Berlin · Hauptstadt der DDR
15. - 16. Dezember 1988

KAMMER DER TECHNIK

Fachverband
Elektrotechnik

# KURZREFERATE

# Fachtagung „Computer- und Mikroprozessortechnik '88"

mit internationaler Beteiligung

Kammer der Technik
-Präsidium-
Fachverband Elektrotechnik

# K U R Z R E F E R A T E

## Fachtagung
## "Computer- und Mikroprozessortechnik '88"

Anfragen richten Sie bitte an:

Kammer der Technik
Präsidium
Fachverband Elektrotechnik
PSF 1315
Berlin - DDR
1 0 8 6

Tel.: 22 65 285

Als Manuskript gedruckt.
Es konnten nur die Kurzreferate aufgenommen werden, die von den Autoren zur Verfügung gestellt wurden.

Programmablauf

P 1 Industriecomputer - neue Wege in der Automatisierungstechnik

P 2 Zur Architektur von Hard- und Software der 32 bit-Klein- und Mikrorechner des SKR

P 3 Der Bildungscomputer robotron A 5105 - ein leistungsfähiger Personalcomputer für das Bildungswesen der DDR

P 4 Kleincomputer in der DDR - Stand und Tendenzen

A 1 Expertensysteme und Fuzzy Reasoning

A 2 Effektive Mikrorechner - Softwaretechnologie auf FORTH-Basis - ein hochentwickeltes Programmiersystem, Spracherweiterung für künstliche Intelligenz

A 3 Modula-2: Programmiersprache, Werkzeuge und Programmbibliotheken zur Erstellung portabler Automatisierungssoftware auf DDR-Rechnern

A 4 Multitask - Programmierabarbeitung im Betriebssystem CP/M-86

A 5 16 bit-Echtzeit-Multirechnersystem mit E/A-Prozessoren 8089

A 6 EMR-Entwicklungssystem unter SCPX (CPA)

B 1 Aspekte bei Multiprozessorentwicklungen

B 2 Arbeitsweise einer Datenflußarchitektur am Beispiel des uPD 7281

B 3 Zur Entwurfsmethodik von Entwicklungs- und Prozeßrechnersystemen

B 4 Ein hochproduktiver Komplex auf Basis eines Prozessors für Matrizen-Rechnungen und eines 16 bit-Minirechners vom Typ CM 4

B 5 Einbau eines Echtzeit-Zählers und eines Operatiospeichers mit abhängiger Stromversorgung in 16 bit-Personalcomputer und Methoden ihrer Steuerung

B 6 Entwurfssystem für VLSI-Speicher

C 1 Modulares Mikroprozeßrechnersystem S 4000

C 2 Projektierung von Mikroprozessor-Modulen für verteilte Steuerungssysteme von technologischen Ausrüstungen

C 3 Entwicklung von Software für lokale mikroprozessorgesteuerte dynamische Regler

C 4 Entwurfsmethodik und Sprachunterstützung für verteilte Steuerungen in der Nachrichtentechnik

C 5 Universelles Prozeßkoppelmodul

C 6 U 840 PC-Spezialprozessor für binäre speicherprogrammierbare Steuerungen

C 7 Assembler/Interpreter für U 840 PC auf Personalcomputer

C 8 Mikroprozessorsysteme für die Steuerung von technologischen Ausrüstungen

C 9 Bewertung der Produktivität von Mehrprozessor-Steuerungseinrichtungen

D 1 Einführung wissensbasierter Systeme zur Fehlerlokalisierung in komplexe n programmierbaren Schaltungen (Objekt Schaltkreistester)

D 2 ARGUS - flexibles Testsystem für verteilte Steuerungen

D 3 Eine Näherung an den Paralleltest von peripheren Geräten

D 4 16 bit-Prüfrechner-System (Prüfung von bestückten Leiterkarten)

D 5 Einige Optimierungsprobleme der Stoßdämpfung der mechanischen Konstruktion von Industriecomputern, die in dynamischer Umgebung arbeiten

D 6 Fragen der mechanischen Konstruktion von Industriecomputern

D 7 Steuerungskonzept eines modularen Logikanalysesystems

D 8 Logikanalysator LA 100 -
Bestandteil eines modularen Logikanalysesystems

D 9 Der Heimcomputer als Hilfsmittel bei der Übersetzung

Ersatzvortrag

E 1 Development Tools for Single-Chip Microcomputers

# ZUR ARCHITEKTUR VON HARD- UND SOFTWARE FÜR 32 bit KLEIN- UND MIKRORECHNER DES SKR

Horn, Th., Prof. Dr. sc. techn., KDT, Dresden, DDR

In der gegenwärtigen Etappe der weiteren Intensivierung der Volkswirtschaft hat insbesondere der breite Einsatz von 32-Bit-Klein- und Mikrorechnern auf den Gebieten der Schlüsseltechnologien eine wachsende Bedeutung. Aus diesem Grund wird in allen Ländern des RGW erfolgreich an der weiteren Entwicklung des Systems der Kleinrechentechnik (SKR) gearbeitet.
Die Leistungsparameter der bisher entwickelten und in die Produktion übergeleiteten 32-Bit-Rechenanlagen zeigt Tabelle 1.

Tabelle 1: Leistungsparameter ausgewählter 32-Bit-Rechenanlagen

| Rechenanlage | Elektronika-82 | SM-1700 | SM52/12 | K1840 | Elektronika MS 0104 |
|---|---|---|---|---|---|
| Herstellerland | UdSSR | UdSSR | CSSR | DDR | UdSSR |
| Verarbeitungsbreite [Bit] | 32 | 32 | 32 | 32 | 32 |
| Befehlssatz | 248 | 248 | 248 | 304 | 178 |
| Registersatz | 16 | 16 | 16 | 16 | 16 |
| Verarbeitungsleistg. [$10^6$ Bef./s] | 0,6 | 0,3 | 1,0 | 1,0 | 0,3 |
| Physischer Adreßbereich [MByte] | 16 | 16 | 1 GByte | 1 GByte | 4 |
| Hauptspeicher [MByte] min. | 4 | 4 | 8 | 8 | 1 |
| max. | 16 | 16 | 32 | 32 | 4 |
| Max. Datendurchsatz [MByte/s] | 5 | 1,5 | 13,3 | 13,3 | 3 |
| SKR-Bussysteme | 1 | 1 | $\leq 4$[1] | $\leq 4$[1] | |
| MS-Bussysteme | $\leq 2$ | - | $\leq 3$[1] | $\leq 3$[1] | |
| Elektronika-Bus | - | - | - | - | 1 |

[1] Zur Grundausstattung gehört 1 SKRBUS-Adapter. Insgesamt 4 Adapter (SKRBUS oder MSBUS) möglich.

Von vorrangiger Bedeutung für den Einsatz in der DDR ist in erster Linie der vom VEB Kombinat Robotron entwickelte und produzierte 32-Bit-Kleinrechner K1840. Die Hardwarestruktur des K1840 ist in Bild 1 dargestellt. Wesentliche Bestandteile sind vor allem die 32-Bit-CPU K2810 mit dem Gleitkommabeschleuniger K2812, die zusammen die in Tabelle 2 aufgeführten Datenformate realisieren. Die CPU wird über ein Konsolensubsystem KSS K2811 auf Basis eines K1630-Prozessors bedient. Über das Konsolensubsystem werden auch Mikroprogramme und die Testsoftware geladen. Der Datenaustausch mit dem Prozessor erfolgt über den Internen Synchronbus (ISB) mit einer hohen Datenübertragungsrate von max. 13,3 MByte/s.

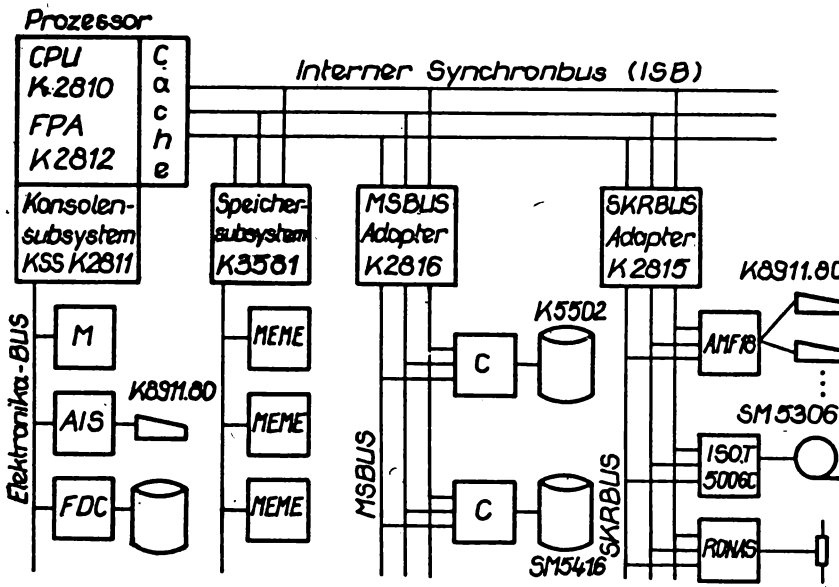



Bild 1: Hardwarestruktur des K1840

Zur Reduzierung der Speicherzugriffe werden Daten und Befehle in einem Cache-Speicher mit 8 kByte Speicherkapazität nach einem halbassoziativen Prinzip zwischengespeichert. Das Speichersubsystem dient zum Anschluß von max. 16 Speichersteckeinheiten. Der Anschluß der Peripherie erfolgt entweder über den MSBUS-Adapter für die Massenspeicher oder über den SKRBUS-Adapter für die klas-

sische Peripherie, wie auch bei den 16-Bit-SKR-Rechnern verwendet. Maximal können insgesamt vier Bus-Adapter angeschlossen werden, wobei in der Regel ein MSBUS-Adapter und ein SKRBUS-Adapter zur Grundausstattung gehören.

Tabelle 2: Datenformate der 32-Bit-Prozessoren

| | | |
|---|---|---|
| **Ganze Zahlen (integer)** | | |
| Byte | 8 Bit | -128 - +127 |
| Wort | 16 Bit | -32768 - +32767 |
| Langwort | 32 Bit | -2147483648 - +2147483647 |
| **Natürliche Zahlen (unsigned integer)** | | |
| Byte | 8 Bit | 0 - 255 |
| Wort | 16 Bit | 0 - 65535 |
| Langwort | 32 Bit | 0 - 4294967295 |
| Quadwort | 64 Bit | 0 - $1{,}8 \cdot 10^{19}$ |
| Octawort | 128 Bit | 0 - $3{,}39 \cdot 10^{38}$ |
| **Gleitkommazahlen** | | |
| F-Format | 32 Bit | $+0{,}29 \cdot 10^{-38}$ - $+1{,}7\ 10^{38}$ |
| D-Format | 64 Bit | $+0{,}29 \cdot 10^{-38}$ - $+1{,}7\ 10^{38}$ |
| G-Format | 64 Bit | $+0{,}56 \cdot 10^{-308}$ - $+0{,}90\ 10^{308}$ |
| H-Format | 128 Bit | $+0{,}84 \cdot 10^{-4932}$ - $+0{,}59\ 10^{4932}$ |
| Zeichenketten | 0 - 65535 Byte | |
| gepackte Dezimalzahlen | 0 - 16 Byte (31 Ziffern und Vorzeichen) | |
| Bitfelder variabler Länge | 0 - 32 Bit | |

Wesentliches Merkmal der 32-Bit-Architektur ist die virtuelle 32-Bit-Adresse, die die Adressierung eines virtuellen Adreßraumes von max. 4 GByte gestattet. Die Aufteilung des virtuellen Adreßraumes in 4 Regionen und ihre Verwendung zeigt Bild 2. Aufgrund des begrenzten physischen Speichers können von jedem Prozeß nur Teile zur Abarbeitung in den Hauptspeicher aufgerufen werden. Unter einem Prozeß wird die Verwaltungseinheit eines in Abarbeitung befindlichen Programms verstanden. Die Programme werden zu ihrer Verwaltung in Seiten eingeteilt, wobei eine Seite einem Plattenblock (512 Byte) entspricht. Die Verwaltung von Seiten erfolgt auf der Grundlage des Paging-Verfahrens. Die im Hauptspeicher befindliche Teilmenge von Seiten eines Prozesses ist die Arbeitsmenge (Working Set). Während der Abarbeitung der Seiten, die sich im physischen Speicher befinden, macht sich eine dynamische Adreßumsetzung erforderlich, bei der die virtuelle Seiten-

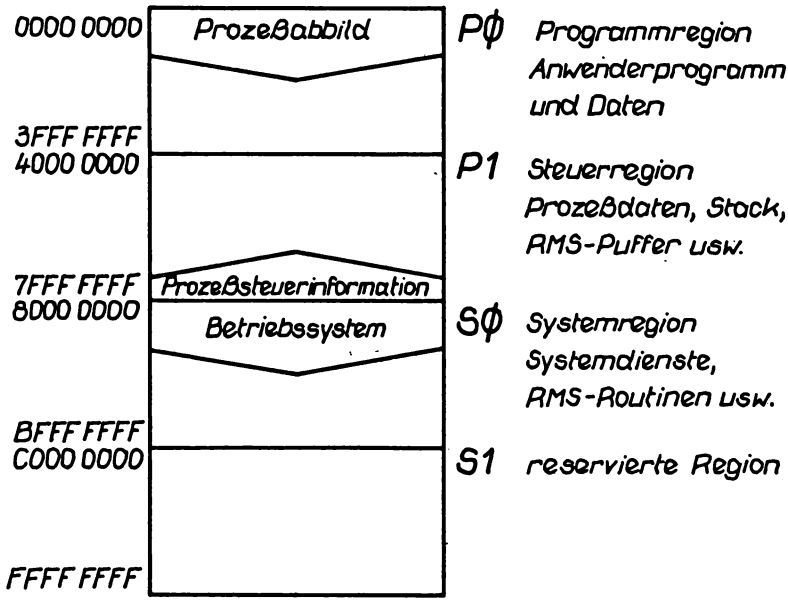



Bild 2: Aufteilung des virtuellen Adreßraumes

nummer VPN in eine physische Seitennummer PPN umgewandelt wird. Die Umwandlung einer virtuellen Systemadresse zeigt Bild 3, wobei sich die Seitentabelle für die Systemregion SØ auf einem zugeordneten physischen Adreßbereich befindet. Für die Systemregionen PØ und Systemregionen P1 der einzelnen Prozesse befinden sich die Seitentabellen in der Systemregion SØ, so daß der in Bild 3 dargestellte Ablauf für die Nutzeradressen aus PØ und P1 zweistufig abläuft. Zur Reduzierung des Zeitaufwandes für die dynamische Adreßumsetzung verfügt der K1840 über einen Translationspuffer, der nach einem assoziativen Prinzip bis zu 128 Adreßumsetzungen speichern kann.

Als Basisbetriebssystem für den CAD/CAM-Einsatz kommt das Betriebssystem SVP-1800 (MOS VP) als System zur Verwaltung von Virtuellen Prozessen zum Einsatz. Das SVP-1800 ist ein universelles prioritätsgesteuertes Time-Sharing-Betriebssystem, das für ein breites Einsatzspektrum von Systemen der Prozeßautomatisierung

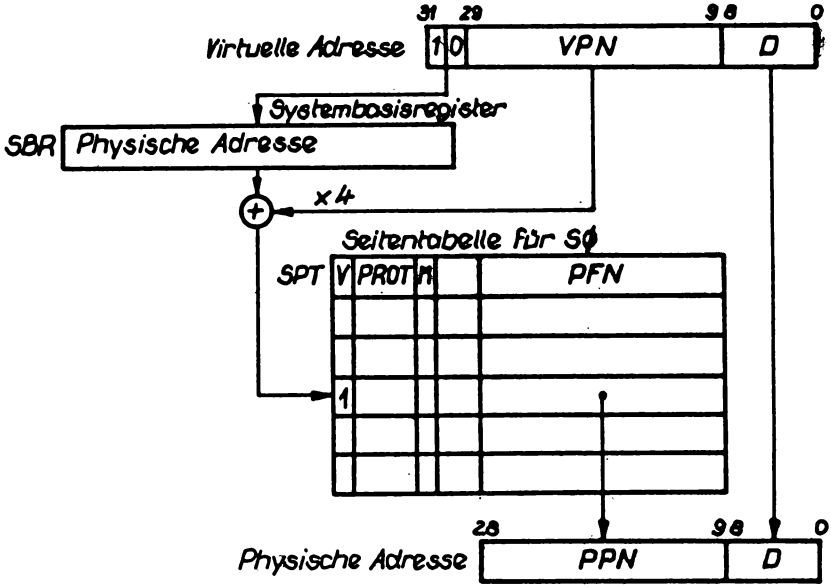



Bild 3: Umrechnung einer virtuellen Systemadresse

bis hin zu großen Time-Sharing-Systemen mit mehreren Dutzend Terminals geeignet ist.
Die prinzipielle Struktur des SVP-1800 ist in Bild 4 dargestellt, wobei Schicht Ø die reale Maschine ist. Schicht 1 ist der Betriebssystem-Kern, der für die unmittelbare Verwaltung der Hardwarekomponenten zuständig ist. Schicht 2 sind die Systemdienste, die für die Anwenderprogramme komplexe Dienstleistungen wie logische E/A, Zeitdienste, Ruf zur Prozeß- und Speicherverwaltung usw. zur Verfügung stellen. Schicht 3 bietet mit dem Satzzugriffssystem RMS (Record Management System) ein komfortables Filesystem, das u. a. hierarchische Fileverzeichnisse und die Fileorganisationsformen sequentiell, relativ und index-sequentiell realisiert. Schicht 4 ist der Kommandointerpreter SCL, der eine moderne flexible Bedienersprache implementiert, die die interaktive und stapelorientierte Verarbeitung von Kommandoprozeduren zuläßt.

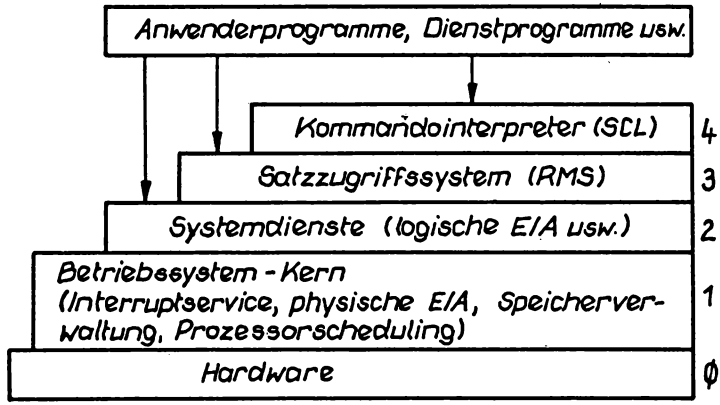



Bild 4: Schichtenmodell des Betriebssystems SVP-1800

Auf dem Nutzerniveau wird neben den Anwenderprogrammen die gesamte weitere Standardsoftware wie Dienstprogramme, Compiler, Linker usw. abgearbeitet.
Zur Realisierung eines effektiven Datenschutzes und der Gewährleistung einer hohen Systemsicherheit werden für alle beanspruchten Ressourcen Kontingente eingeführt, die die Ressourcennutzung limitieren. Ein umfangreiches abgestuftes Privilegiensystem gestattet eine optimale Systemsicherheit, da dem Nutzer für seine durchzuführenden Arbeiten differenzierte Privilegien zugeteilt werden können. Der Datenträger- und Dateischutz wird auf der Grundlage eines Nutzeridentifikationskodes gewährleistet, der eine Gruppeneinteilung der Nutzer vorsieht. Für besonders zu schützende Objekte (Geräte, Datenträger, Files etc.) können spezielle Zugriffssteuerlisten benutzt werden, die für jeden Nutzer konkret die Zugriffsrechte regeln. Für die Nutzer von 16-Bit-Rechenanlagen des SKR unter den Betriebssystemen OS-RW bzw. OMOS-1600 sieht das SVP-1800 einen speziellen Kompatibilitätsmodus vor, der bei geringfügigen Einschränkungen einen problemlosen Übergang von der 16- zur 32-Bit-Technik gestattet. Der Kompatibilitätsmodus kann auch für eine effektive Programmentwicklung der Software für die 16-Bit-Technik genutzt werden.

Der Bildungscomputer robotron A 5105 - ein leistungsfähiger Personalcomputer für das Bildungswesen der DDR

Erler, W., Dr.-Ing., Dresden, DDR

Im VEB Robotron-Meßelektronik "Otto Schön" Dresden (VEB MKD) erfolgte die Entwicklung des Bildungscomputers A 5105. Die ersten Geräte zur Vorbereitung der Computerausbildung wurden am 30. 9. 1988 an die Partner des Ministeriums für Volksbildung; des Ministeriums für Hoch- und Fachschulwesen und des Staatssekretariates für Berufsbildung übergeben.

Die Entwicklung basiert auf einem Pflichtenheft, das gemeinsam durch den VEB MKD, die Akademie der Pädagogischen Wissenschaften im Auftrage des Ministeriums für Volksbildung und des Staatssekretariates für Berufsbildung erarbeitet worden ist. Das Pflichtenheft entspricht den Forderungen, die international an einen Bildungscomputer gestellt werden.

Schwerpunkte des Forderungsprogrammes des Bildungswesens waren:

- Kompaktausführung: stabil, robust, formschön,
- ergonomische Gestaltung von TASTATUR und Monitor,
- 1 Diskettenlaufwerk als Massenspeicher,
- 64 kB interner Arbeitsspeicher,
- Punktgrafik,
- Farbbildausgabe,
- Anschluß Schülerexperimentiergerät und Modulsteckplätze,

Zusatzforderungen:

- wahlweiser Anschluß von KMBG,
- Steuerhebelanschluß,
- Monitor mit Bildschirmdiagonale $\leq 31$ cm,
- Netzverkopplung für Computerkabinette.

2.

Der Bildungscomputer besteht konstruktiv aus den drei Grundelementen:

Computergrundgerät (CGG)
Diskettenspeichereinheit (DSE) K 5601
Monitor K 7222.23

Diese Grundelemente sind elektrisch und mechanisch miteinander verkoppelt.

3.

Die Entwicklungskonzeption geht davon aus, das CGG so zu gestalten, daß es auch separat mit einem eigenen Netzteil betrieben werden kann. In diesem Falle ist der Anschluß eines Fernsehgerätes als Monitor und eines Kassettenmagnetbandgerätes als Massenspeicher möglich. Damit besteht die Möglichkeit, daß durch den Lehrer der Unterricht zu Hause vorbereitet werden kann; darüber hinaus ist damit der Hauptbestandteil für ein neues Heimcomputersystem verfügbar.

Um eine hohe Leistungsfähigkeit des Computers zu erreichen bei geringem Gewicht und geringem Volumen wurde die Verwendung hochintegrierter Schaltkreise (z. B. RAM-Speicher 256 kbit) vorgesehen und der Einsatz von zwei Kundenwunsch-Schaltkreisen auf Standardzellenbasis (Speicherverwaltung, Videointerfacecontroller). Als Mikroprozessor ist der Schaltkreis U 880 eingesetzt.

4.

Als Betriebssystem sind vorgesehen

RBASIC und
SCP

Dabei ist der BASIC-Interpreter im Gerät implementiert in einen

32 kByte ROM

Ein 16 kByte ROM enthält die RBASIC-DISK-Erweiterung. Das

SCPX 5105

RAM-Betriebssystem wird mit Systemdiskette geladen; es ist kompatibel zu dem SCP des Personalcomputers PC 1715 des VEB BWS. Dadurch ist die Standardsoftware des SCP wie Redabas KP, Turbo-Pascal usw. nutzbar.
Das RBASIC enthält ca. 150 Kommandos, Anweisungen und Funktionen; es ist zum BASI des Eser-PC EC 1834 aufwärtskompatibel (Bildungscomputerprogramme laufen auf dem 1834!).

## 5. Technische Daten

COMPUTERGRUNDGERÄT

| | |
|---|---|
| Mikroprozessor | UA 880 D |
| Taktfrequenz | 3,75 MHz |
| Speicherkapazität | 32 kbyte ROM (Geräte-Treiber, BASIC-Interpreter, 64 kbyte RAM (Arbeitsspeicher), 64 k x 16 bit (Video-RAM) |
| Speichererweiterung | z. B. 64 kbyte RAM durch Ergänzungsmodul |
| Tastatur | Computerflachtastatur mit 40 alphanumerischen Tasten, 5 Kursortasten, 11 Steuertasten, 5 freiprogrammierbaren Funktionstasten |
| Anzeige | - MONITOR robotron K 7222.23<br>- Spezial-Farbmonitor<br>- handelsübliches Fernsehgerät (über RGB-Eingang oder HF-Modulator) |
| Bildaufbau | Text - 25 Zeilen x 40 Zeichen (max. 16 Seiten)<br>- 25 Zeilen x 80 Zeichen (max. 8 Seiten)<br>Grafik - 320 x 200 Bildpunkte (max. 6 Seiten)<br>- 640 - 200 Bildpunkte (max. 3 Seiten) |
| Farbausgabe | Text - 16 Vordergrundfarben<br>8 Hintergrundfarben<br>Grafik - 4 aus 16 Farben je Bildpunkt für 640 x 200 (Farbausgabe nur mit Farbmonitor bzw. RGB-Anschluß möglich)<br>16 Farben je Bildpunkt für 320 x 200 |
| Zeichenvorrat | 256 Zeichen (alphanumerische Zeichen, Sonderzeichen bzw. Grafiksymbole) |
| Tonausgabe | über Lautsprecher in der DISKETTENSPEICHER-EINHEIT oder angeschlossenes Fernsehgerät |
| Software | RBASIC (in 32-kbyte-ROM) |

DISKETTENSPEICHEREINHEIT

| | |
|---|---|
| Massenspeicher | - Diskettenspeicher robotron K 5001 mit Laufwerk 1.6 für 5,25 Zoll-Disketten (max. 800 kbyte)<br>- Diskettenansteuerung mit Floppy-Disk-Controller U 8272 |
| Lautsprecher | Typ L 1001 (über Steuerbefehl abschaltbar) |
| Software | RBASIC-Disk-Erweiterung (in 16-kbyte-ROM) |
| Netzteil | Zentrale Stromversorgung für CGG, DSE und MONITOR |
| Interfaces | siehe 6. |

## 6. Anschlußmöglichkeiten für periphere Geräte und Erweiterungsmodule

- 2 Bus-Steckplätze: Modulsteckplatz im CGG und BUS-Anschluß an der DSE,
- 2 E/A-Buchsen mit jeweils 8 programmierbaren E/A-Kanälen und je einem CTC-E/A-Kanal,
- 1 RGB-Anschluß für Farbmonitor oder FFS,
- 1 Anschluß für Diskettenbeistellung (max. 2 Laufwerke).
- 1 Anschluß für Vernetzung.
- Video- und Audio-Anschluß.
- 2 Buchsen für Steuerhebel (am CGG).
- 1 Anschluß für Kassettenrecorder (am CGG)
- Standard-Interface für Drucker + Schreibmaschine (Basis V 24) (auch an CGG. zusätzlich Centronix-Schnittstelle) und getrennter Plotter-Anschluß (V 24) an DSE.

## 7. Tastatur

In Auswertung der nicht voll befriedigenden Lösung der Tastatur des Kleincomputers KC 85/87 wurde für den BIC/CGG eine Schreibmaschinentastatur (Flachtastatur) vorgesehen, die in der Betätigung keine Wünsche offenläßt.
Sie hat 40 alphanumerische Tasten, 5 Kursortasten, 11 Steuertasten und 5 freiprogrammierbare Funktionstasten. Insgesamt lässen sich 287 verschiedene Zeichen eingeben; diese sind nach dem ASCII-Code codiert. Insbesondere für mathematisch-physikalische Programme sind die Sonderzeichen. wie z. B.

$\mathcal{L}, \beta, \mathcal{I}, \pi, \Sigma, \sigma, \mu, \tau, \varphi, \theta, \Omega, \sigma, \varepsilon, \lambda$

vorteilhaft.

## 8. Lokales Netz

Der BIC ist nicht nur für Einzelanwendung konzipiert, sondern auch für den Aufbau von Computerkabinetten, mit in sich vernetzten Geräten.
Die Kopplung erfolgt mit Koax-Leitung 75Ω ; die Gesamtlänge kann bis 100 m betragen und erfolgt über entsprechende Anschlußdosen.
Durch die Kopplungsmöglichkeiten kann z. B. vom Lehrerplatz ein Zugriff auf die Schülerarbeitsplätze erfolgen.

## 9. Diskettenbetrieb

Es ist ein Diskettenlaufwerk 1.6 eingesetzt; es arbeitet vorzugsweise mit 5 1/4"-Diskette mit Doppeldichte und doppelseitig beschreibbar. Sie haben formatiert eine Speicherkapazität von 780 kByte.
Obwohl nur 1 Laufwerk im BIC enthalten ist, lassen sich Disketten kopieren; dabei ist die kopierbare Datenmenge durch den Speicherinhalt des Computers begrenzt. Es ist die Verarbeitung von sequentiellen und von Direktzugriffsdateien möglich.

Sequentielle Dateien gestatten eine maximale Ausnutzung des Diskettenspeichervermögens; die eingegebenen Daten werden fortlaufend aneinandergereiht eingelesen, nur mit zwischengeschaltetem "Enter" (Ø D) bzw. Zeilenendzeichen (Ø A).

Beim Arbeiten mit der Datei muß stets die vollständige Datei gelesen werden; einzelne Daten lassen sich erst dann aufrufen; wenn sie geändert werden, muß die gesamte geänderte Datei wieder auf Diskette geschrieben werden. Bei Direktzugriffsdateien muß für jeden Satz beim Eröffnen der Datei die Satzlänge festgelegt werden; sie ist dann für jeden Satz gleich. Wenn weniger Daten eingelesen werden, wird der Rest mit Leerzeichen aufgefüllt; bei längeren Eingaben werden die Zeichen nach Satzende abgeschnitten. Sätze können einzeln gelesen geändert und wieder eingeschrieben werden.

## 10. Bildausgabe

- Der Bildspeicher hat eine Kapazität von 64 K x 16 bit im Video-RAM. Dabei sind 16 K x 16 bit für Text-Screens vorgesehen, das sind 16 bzw. 8 Bildschirmseiten für die Modi 40 bzw. 80 Zeichen pro Zeile. 48 K x 16 bit sind für Graphik-Screens, das sind 6 bzw. 3 Bildschirmseiten für Screen 2

(320 x 200 Bildpunkte) bzw. Screen 3 (640 x 200 Bildpunkte). Die einzelnen Seiten lassen sich in einer Zeit von 20 ms lesen, so daß schnell bewegte Bilder erzeugt werden können.

- Window-Anweisungen für Text und Graphik sind vorgesehen zur mulivalenten Nutzung von Bildschirmabschnitten.

- Programmänderungen/Textänderungen sind durch den Screen-Editor möglich, dabei können Änderungen über Kursorbewegungen auf dem gesamten Bildschirm erfolgen.

## 11. Besonderheiten

Bei Betrieb des CGG mit dem KMBG "LCR" vom VEB Mikroelektronik Gera läßt sich das KMBG vom Computer aus steuern: Files werden selbsttätig gesucht und eingelesen; dadurch ist ein Betrieb wie mit Diskette möglich, der nur etwas langsamer vor sich geht.

KLEINCOMPUTER IN DER DDR - STAND UND TENDENZEN

Domschke, W. Dr.-Ing. KDT Mühlhausen DDR

## 1. Einführung

Die Fortschritte der modernen Mikroelektronik, insbesondere die ständige Leistungssteigerung und die Preissenkungen der Bauelemente, machte die Entwicklung und Produktion von relativ preiswerten Computern mit abgestuften Leistungsparametern möglich. Speziell für die Ausbildung und den Heimbereich steht die Klasse der Kleincomputer zur Verfügung.
Ausgehend von einer kurzen Darstellung der derzeitig produzierten und in Entwicklung befindlichen Kleincomputer sollen im folgenden anhand der wesentlichen Einsatzgebiete dieser Technik Anforderungen erarbeitet werden, die für eine breite Annahme der Geräte durch die Anwender erfüllt werden müssen, um daraus Tendenzen für die weitere Entwicklung von Kleincomputern abzuleiten.

## 2. Kleincomputer in der DDR

Derzeitig werden in der DDR drei Kleincomputersysteme industriell gefertigt, drei weitere sind in der Entwicklung. Im folgenden werden diese kurz vorgestellt.

Computerbausatz Z1013

Hersteller: VEB Robotron Elektronik Riesa

Dieser Bausatz ist vor allem für die Einarbeitung in die Maschinenprogrammierung des Prozessors U880 geeignet.

Kleincomputer KC85/3

Hersteller: VEB Mikroelektronik Mühlhausen

Dieser Kleincomputer wird zur Informatikausbildung in der Berufsschule, in der polytechnischen Ausbildung, in Computerclubs, in der Industrie, Landwirtschaft, Handwerk und im Heimbereich eingesetzt. Er ist geeignet für Standardaufgaben, zum Erlernen der Programmierung sowie zum Spielen.

Kleincomputer KC87

Hersteller: VEB Robotron Meßelektronik Dresden

Dieser Kleincomputer ist im wesentlichen in Computerkabinetten eingesetzt und wird zur polytechnischen Ausbildung, in Arbeitsgemeinschaften und im Heimbereich genutzt. Er ist geeignet zum Erlernen des Umgangs mit dem Computer, der Programmierung sowie zum Spielen.

In Entwicklung befinden sich:

Kleincomputer KC85/4

Entwickler: VEB Mikroelektronik Mühlhausen

Zusatzbaugruppen: wie KC85/3, weiterhin KC-Floppy mit CP/M-kompatiblen Betriebssystem (in Entwicklung),

Dieser KC wird den KC85/3 ablösen. Er ist sowohl in der Hardware als auch in der Software kompatibel zum KC85/3 und KC85/2. Neben wesentlicher Überarbeitung des Betriebssystems ist die Farbauflösung für jeden Bildpunkt möglich und es können zwei voneinander unabhängige Bilder gespeichert werden.

Bildungscomputer A5105

Entwickler: VEB Robotron Meßelektronik Dresden

Der Bildungscomputer wird nach Vorgaben der Volks- und Berufsbildung entwickelt und soll ab 1989 eingesetzt werden. Als Betriebssystem wird SCPX verwendet, das kompatibel zum SCPX des PC1715 ist.

Kleincomputer KC90

Entwickler: VEB Kombinat Robotron

Der KC90 soll als Grundgerät des A5105 als Heimcomputer in den Handel gebracht werden.

Kennzeichnend für alle die vorgestellten Kleincomputer ist der Prozessor U880 mit einer Verarbeitungsbreite von 8 Bit.

## 3. Anforderungen an Kleincomputer

### 3.1. Einsatzgebiete

Kleincomputer haben aufgrund ihrer Leistungsmerkmale spezielle Einsatzgebiete. Im wesentlichen werden sie genutzt

- zum Erlernen des Umgangs mit dem Computer,
- zum Erlernen der Grundlagen der Programmierung,

- für einfache Abrechnungsaufgaben, Textverarbeitung, Bürorationalisierung,
- für einfache Steuerungs-, Regelungs- und Registieraufgaben,
- zur Demonstration industrieller Computeranwendung (z.B. CAD),
- zum Spielen usw.

Alle diese Beispiele stellen andere Anforderungen an die Leistungsfähigkeit und den Ausrüstungsgrad der KC.
Um möglichst ein breites Einsatzspektrum der KC zu erreichen, werden nachfolgend Forderungen für die künftige Gestaltung der KC aufgelistet.
Bei der Textverarbeitung und den Spielen kann aus den bisherigen Einsatzerfahrungen geschlußfolgert werden, daß die betrachtete Geräteklasse im heutigen Ausrüstungsgrad im wesentlichen in der Lage ist, die Anforderungen zu erfüllen, was bei CAD wegen der zu geringen Arbeitsgeschwindigkeit und des zu geringen Speicherplatzes nicht der Fall ist. Deshalb wurde in den nachfolgenden Betrachtungen nur die Demonstration der wesentlichsten Elemente von CAD zu Ausbildungszwecken herangezogen.

## 3.2. Mensch-Maschine-Interface

Ein Kleincomputer, der universell anwendbar sein soll, muß über eine Schreibmaschinentastatur mit einem gesonderten Cursortastenfeld und einem "Zehnerblock" verfügen. Der Bildschirm muß grafikfähig mit mindestens 8 Farben und einer Auflösung von mindestens 512 x 200 Bildpunkten sein. Damit können problemlos 64 Zeichen / Zeile dargestellt werden und es können noch handelsübliche Fernsehgeräte mit RGB-Eingang verwendet werden. Für viele zukünftige Anwendungen und wegen Fragen der Kompatibilität mit größeren Computern sollte die Auflösung des Grafikbildschirmes 640 x 200 Bildpunkte betragen bei der Möglichkeit der Änderung des Formats durch Programmierung.
Am Kleincomputer sollten Standardschnittstellen für den Anschluß von Peripheriegeräten vorhanden sein oder nachgerüstet werden können (serielle und parallele).

## 3.3. Speicher

Der Kleincomputer muß einen ROM enthalten, in dem die Firmware für die Realisierung der wichtigsten Funktionen, wie z.B. Tastaturabfrage, Bildschirmansteuerung, Steuerung des externen Speichers (RAM-Disk, Kassettenmagnetbandgerät, Floppy-Disk) enthalten ist. Weiterhin sind mindestens 64 KByte RAM zu installieren, entsprechend dem heutigen Stand der Entwicklung sollten 256 KByte RAM vorgesehen werden. Für die meisten der betrachteten Anwendungsfälle der 8 Bit-Technik reicht ein Kassettenrecorder mit einer Datenübertragungsgeschwindigkeit von mindestens 3000 Bit/s als externer Speicher, wenn eine RAM-Disk vorhanden ist. Damit läßt sich ein System aufbauen, mit dem Standardbetriebssysteme und -anwendersoftware nutzbar sind und das billiger ist als Personalcomputer mit Floppy-Disk.
Für Prozessoren mit einer größeren Verarbeitungsbreite (16 Bit) müssen mindestens 512 KByte RAM enthalten sein. Als externer Speicher kann nur ein Floppy-Disk die Anforderungen erfüllen.

## 3.4. Prozessor

Von den einzelnen Anwendungsfällen werden unterschiedliche Anforderungen an den Prozessor gestellt. In den nächsten Jahren sind die 8-Bit-Prozessoren in der DDR noch favorisiert wegen des geringeren Preises. Sollen die Anforderungen an die Arbeitsgeschwindigkeit erfüllt werden, sind 8-Bit-CPU mit höherer Taktfrequen (4 MHz und mehr) und Coprozessoren, wie z.B. Videoprozessor und Floppy-Disk-Controler, einzusetzen. Perspektivisch werden auch bei Kleincomputern die 16-Bit-Prozessoren angewendet. Anhand eines Beispieles soll das verdeutlicht werden.
Um das Bild eines Luftballons in der Größe 32 x 32 Bildpunkte bei einer Grafikauflösung von ca. 82000 Bildpunkten kontinuierlich in 3 Sekunden quer über den Bildschirm zu bewegen, muß aus der jeweiligen Position die zugeordnete Speicheradresse des Bildwiederholspeichers berechnet werden, der dort befindliche Bildausschnitt von 32 x 32 Bildpunkten in den Arbeitsspeicher übernommen werden und das Bild des Luftballons eingeblendet

werden.
Nach Ablauf der Bildphase (nach ca. 40 ms) muß der "gerettete" Bildausschnitt zurückgespeichert werden, und der Vorgang beginnt von vorn.
Daraus ergeben sich für die unterschiedlichen Prozessoren und Taktfrequenzen folgende Rechenzeitbilanzen:

| Prozessor | Taktfrequenz | Coprozessor | Rechenzeit[1]) |
|---|---|---|---|
| U8880 | 1.75 MHz | keiner | 78% |
| UA880 | 4 MHz | keiner | 34% |
| U8400 | 8 MHz | keiner | 17% |
| UA880 | 4 MHz | GDC[2]) | 28% |
| U8880 | 2 MHz | VDP[3]) | 8% |
| UA880 | 4 MHz | VDP[3]) | 4% |
| 8086 | 2 MHz | keiner | 14% |
| 8086 | 4.77 MHz | keiner | 6% |
| 80286 | 4.77 MHz | keiner | 1% |

[1]) für die Realisierung der Bewegung, Abschätzung
[2]) Graphic Display Controler
[3]) Video Display Processor mit Sprites-Einblendmöglichkeit

## 4. Schlußfolgerungen

Kleincomputer bilden wegen ihres Preises, ihres abgestuften Leistungsvermögens, der Nutzung von möglichst vorhandener Peripherie (z.B. Fernsehgerät) und der universellen Erweiterungsmöglichkeit eine eigene Klasse. In der DDR werden derzeit drei Geräte produziert und drei weitere befinden sich in Entwicklung. Kennzeichnend für diese Kleincomputer ist die 8-Bit-CPU.
Für viele Aufgaben sind diese KC optimal nutzbar, bei einer Reihe von Anwendungen reicht die Leistungsfähigkeit nicht aus. Anhand von ausgewählten Anwendungsfällen wurden Anforderungen erarbeitet, die diese KC erfüllen müssen, wenn sie von den Anwendern auch zukünftig akzeptiert werden sollen. Diese Anforderungen können in den nächsten Jahren auch noch von KC mit einer 8-Bit-CPU erfüllt werden. Ausschlaggebend dafür ist der Preis. Dabei sind folgende Leistungsparameter anzustreben:

| | |
|---|---|
| Prozessor: | UA880D |
| Coprozessoren: | Video-Display-Prozessor<br>(ev. Floppy-Disk-Controler) |
| Taktfrequenz: | 4 MHz |
| RAM: | 64 KByte |
| ROM: | 32 KByte mit Standardbetriebssystem<br>(z.B. CP/M) |
| IRM: | 64 KByte |
| RAM-Disk: | 256KByte |
| Zeichengenerator: | 2 KByte ROM bzw. Softwarerealisierung |
| Display: | Fernsehgerät / Monitor |
| Darstellung: | 80 Zeichen/Zeile, 24 Zeilen |
| Grafik: | 620 x 200 Bildpunkte, andere Formate<br>programmierbar, z.B. 320 x 256 Bildpunkte |
| Ext. Speicher: | Kassettenmagnetbandgeät mit mind. 4500 Bit/s |
| Schnittstellen: | V24, IFSS (nachrüstbar),<br>Centronics (nachrüstbar) |

Mit der Verfügbarkeit eines 16-Bit-Prozessors mit geringem Preis wird dieser auch in die Kleincomputer eingesetzt werden. Somit erreichen die KC die Leistungsfähigkeit von heutigen Personalcomputern. Der Unterschied wird im wesentlichen in den externen Speichern (z.B. Festplatte), in der Grafikauflösung, in der RAM-Kapazität, in der Arbeitsgeschwindigkeit (Taktfrequenz) und den Peripheriegeräten zu finden sein.

# EFFEKTIVE MIKRORECHNER-SOFTWARETECHNOLOGIE AUF FORTH-BASIS: EIN HOCHENTWICKELTES PROGRAMMIERSYSTEM

Karadshow, L., Dipl.-Ing., NILES, Berlin, TH Ilmenau, DDR

## 1. EINFÜHRUNG

Die heutige Rechnerexplosion stellt qualitativ neue Forderungen an die Technologie der Softwareentwicklung. Als hochkomfortable, leistungsstarke Programmiersysteme werden komplizierte, teuere Softwarepakete geschaffen, die nicht flexibel genug, untransparent und meist maschinenabhängig sind. Weiterhin schließen sich Programmierkomfortabilität und Systemzugänglichkeit weitgehend gegenseitig aus. Ausgebaute Programmiersysteme bieten keine ausreichenden Möglichkeiten der nutzerspezifischen Erweiterung zu einer beliebigen Fachsprachumgebung. Es muß deswegen oft bei der Programmentwicklung auf eine problemangepaßte Sprachunterstützung weitgehend verzichtet werden /1/.
Äußerst problematisch ist die maschinennahe Programmierung. Mächtige Entwicklungssoftware bildet eine wenig durchlässige Schicht zwischen der Hardware und dem Nutzer. Das Ziel, die Maschine möglichst effektiv für die Lösung von Aufgaben einzusetzen, kann nur über umständliche Wege erreicht werden, meist sogar über Kopplung verschiedener Programmierumgebungen, z.B. PROLOG mit C. Es ist selten möglich, erstellte maschinennahe Programmteile zusammen mit High-Level-Programmen komfortabel im Dialog zu testen, damit zu experimentieren.

## 2. DIE FORTH - PHILOSOPHIE

FORTH ist eine neuartige alternative Programmierphilosophie. Es ist eine Programmierumgebung zum *Bau* beliebiger Fachsprachen, in denen sich die Problemauffassungen modellieren und die Lösungen am effektivsten und ellegantesten beschreiben lassen /2/. Die FORTH-Philosophie ist die natürliche praktische Lösung des Problems des effektiven und produktiven Herangehens an neuauftauchende Probleme in der Computerwelt, für die es keine prädestinierte Software gibt. Sie weist Eigenschaften auf, die für die Programmierung bei der gegenwärtigen Rechnerexplosion unabkömlich sind: *Einfachheit*, *Offenheit* und *Einheitlichkeit*. Ein

FORTH-System ist eine Gesamtheit hierarchisch strukturierter kompakter und transparenter Programmelemente einheitlicher Sprache (FORTH) und mit einheitlichen Schnittstellen. Ein völlig offenes Konzept liegt vor: aus einzeln zugänglichen, vom Nutzer definierbaren Programmbausteinen, alle systeminternen und Interface-Routinen eingeschlossen, entstehen vollständig neue, in Funktionsweise, Leistungsparameter und Sprachumfang maximal objekt- bzw. prozeßangepaßte Fertigsysteme. Die gedankliche oder materielle Decomposition einer Programmieraufgabe kann dabei unmittelbar in die Programmstruktur umgesetzt werden.
In FORTH realisierte Assembler mit "High-Level"-Strukturen sind Bestandteile desselben Systems, das Interpreter, Compiler und Sprachkern enthält. Erstellen und Test von Maschinencode ist so mit dem Erstellen und Testen von High-Level-Definitionen vom Vorgang her gleich. Damit ist eine Maschinensprache nur eine Erweiterung des jeweiligen Sprachkonzepts und besitzt zudem eine gewisse Maschinenunabhängigkeit (!).
Das Verbinden der Maschinen- mit der High-Level-Programmierung im einheitlichen System ist ein geeigneter Weg, die FORTH-Strategie für das Erreichen erforderlicher neuer Softwarequalitäten effektiv einzusetzen: hocheffektive und komfortable systemnahe Programmierung als Voraussetzung für bessere Ausnutzung der Hardwaregegebenheiten, flexiblere und vielfältige Optimierung, sowie größere Transportabilität der Programme.
Während höhere Programmiersprachen normalerweise ein geregeltes Netz von Konstruktionen vorsehen, ist FORTH das Konzept eines erweiterbaren Systems, bei dem der Nutzer ein eigenes Vocabular aus Objekten und *Operatoren* aufbaut, wobei er *nutzerspezifische Operatortypen* und Konstruktionen definieren kann.
FORTH bietet gute Möglichkeiten für echte KI-Spracherweiterung. Durch das Verbinden von system- und objektnaher Programmierung mit KI-Komponenten in einem einheitlichen System entstehen einmalige Vorteile für die Programmentwicklung.

## 3. DAS FORTH - PROGRAMMIERSYSTEM

Ein typischer Programmentwicklungszyklus besteht aus Analyse (Determinierung der für die Lösung erforderlichen Funktionen), Design (Wahl des Realisierungsweges angesichts vorhandener Ressourcen und Bedingungen), Implementierung, Optimierung und Test.

Die Softwareentwicklung in FORTH beruht auf dem sog. "iterativen Prinzip". Zunächst werden simple Lösungen separat implementiert und möglichst viele Bedingungen ignoriert. Letztere werden zusammen mit den Erkenntnissen des Tests stufenweise in einem iterativen Prozeß übernommen. Im Gegensatz zur strengen Top-Down-, bzw. Bottom-Up-Methode werden mehrere Experimente innerhalb von Minuten durchgeführt. Das iterative Prinzip ist ein untrennbarer Bestandteil einer praxisorientierten Softwarephilosophie.
Dementsprechend ist auch das vorliegende Entwicklungssystem konzipiert. Es ist das Produkt einer Entwicklung an der TH Ilmenau, die im NILES-Stammbetrieb in Berlin fortgesetzt wurde. Das Programmiersystem ist vollständig in FORTH realisiert. Das Zusammenspiel der einzelnen Entwicklungstools, ihre freie Kombinier- und Aufrufbarkeit im einheitlichen System ergeben sich von selbst. Das ist für die zeitintensive benutzer-interaktive Dialogarbeit im iterativen Prozeß von großer Bedeutung. Der FORTH-Assembler und der erweiterbare Compiler bieten eine durchgängige Unterstützung der kombinierten maschinennahen/High-Level-/objektnahen Programmierung. Hochkomfortabler Vollscreeneditor und Decompiler/Debugger werden von zahlreichen, ebenso interpretativ aufrufbaren Diensten zur Quellcode-, Peripherie- und sonstigen Resourcenverwaltung unterstützt. Die Entwicklungtools sind wegen ihrer modularen FORTH-Struktur in mehreren Stufen ausbaubar.
Zur Unterstützung der Applikationsprogrammierung sind zahlreiche, weitgehend automatisch referenzierbare Module enthalten.
Eine maschinenunabhängige Schnittstelle, die bereits auf niedrigem Niveau und mit wenig Aufwand für mehrere verbreitete Rechnersysteme realisiert ist, dient als Basis für die Implementierung der Entwicklungstools und der meisten Applikationsmodule. Somit ist die erstellte Software voll transportabel, die Realisierung einer wirtschaftlich oder technisch notwendigen Trennung von Entwicklungs- und Laufzeitsystem bei bleibenden Optimierungsmöglichkeiten auf dem Zielsystem demzufolge unproblematisch.
Außer im iterativen Stadium erfolgt eine Optimierung des Fertigprogramms im Prozeß der *Metacompilation.* Der Begriff beschreibt die Compilation eines FORTH-Systems durch ein FORTH-System (Metacompiler) und verdeutlicht, daß es sich ggf. um eine Compiler-Compilation handelt. Der Metacompiler stellt eine selbstständige Umgebung für die Erzeugung von Endcompilaten dar und ist nur annähernd mit einem kommerziellen "echten" Compiler vergleichbar.

Bei der Metacompilation wird das Zielsystem *von Null an* ggf. nutzerspezifisch, durchgängig in FORTH aufgebaut, unter Berücksichtigung aller Rechner-/Prozeßbedingungen und bei *variablen* Optimierungskriterien, so daß *mehrere* Versionen eines Programms erstellt werden können. Der Metacompiler dient auch der Crosscompilation von FORTH-Systemen.

## 4. KI - GESTÜTZTE PROGRAMMENTWICKLUNG

Um bei der steigenden Komplexität der Prozeß- und Rechnerumgebung leistungsfähig und effektiv zu bleiben, müssen moderne Software-Entwicklungssysteme zunehmend computergestützte Programmgenerierung ("automatische Programmierung") anwenden. In FORTH-Programmiersystemen ist die stufenweise Realisierung einer solchen Strategie durch Einbindung von KI-Spracherweiterungen /3/ in der Programmentwicklung mit vergleichbar geringem Aufwand möglich.
Statt Problembeschreibung in deklarativer Sprache wird die Anwenderlösung in einer denkbar objektnahen FORTH-basierten, vorerst virtuellen Sprachumgebung modelliert. Beim Design werden nicht vorhandene Resourcen einbezogen. Allgemeines Ziel ist, den nicht problembezogenen Teil der Implemetierung weitgehend dem Programmiersystem selbst zu übertragen. Die Realisierung wird einerseits über formelle Untersetzung von Hardwarebedingungen und Optimierungskriterien durch "automatische" Selektierung von Modulen und Modulgruppen, andererseits durch Generierung von Assembler- und High-Level-Sequenzen zur optimalen Resourcen- und Parameterverwaltung oder zur Implementierung formell beschriebener Funktionen angestrebt. FORTH bietet die einmalige Gelegenheit, Komponenten der "künstlichen Intelligenz" einzusetzen, die eine echte Erweiterung der Programmiersprache darstellen, folglich reell in der Umgebung der Nutzersoftware wirken.

## Literatur


/1/ Goppold,A.: Trends in der Entwicklung der Programmiersysteme. Vierte Dimension 1 (1985) 2

/2/ Brodie, L.: Thinking FORTH. Prentice-Hall, Inc.

/3/ Noack, K.: Effektive Mikrorechner-Softwaretechnologie Computer- und Mikroprozessortechnik '88, Fachtagung, Berlin.

**EFFEKTIVE MIKRORECHNER-SOFTWARETECHNOLOGIE AUF FORTH-BASIS: SPRACHERWEITERUNG FÜR "KÜNSTLICHE INTELLIGENZ"**

Noack, K., Dipl.-Ing., NILES, Berlin, TH Ilmenau, DDR

## 1. BACKTRACKING IN FORTH

FORTH ist backtrackfähig.
Die sonst äußerst problematische Situationsspeicherung ergibt sich bei FORTH aufgrund der stackorientierten Arbeitsweise durch Stack-Ablage auf einem *Trackstack*.
Backtracking bedeutet, daß ein Programm Verzweigungen enthält, deren Entscheidungskriterien erst im nachhinein gewonnen werden. In Verbindung mit Unifikationsmechanismen ermöglicht es die Programmierung in Logik (PROLOG). Unabhängig davon stellt es innerhalb algorithmischer Denkweise eine außergewöhnliche Art der Alternative dar und begründet sich mit der neben der Arbeit mit "höheren" Sprachen häufig anstehenden Forderung nach Einbeziehung der "künstlichen Intelligenz" in die elementare, z.T. maschinennahe, jedenfalls aber algorithmische Programmierung.
Die dafür erforderlichen Mittel lassen sich unter Verzicht auf ohnehin undurchsichtige Automatismen in FORTH speichergünstig und flexibel realisieren. Ein auf NILES-FORTH /3/ basierendes Programmpaket ermöglicht eine PROLOG-ähnliche Herangehensweise innerhalb der Sprache FORTH. Die Spracherweiterung ist gering gehalten und der Einstiegsaufwand bei Kenntnis der FORTH-Programmierung minimal.
Darüber hinaus weist das Trackstack-Prinzip den Vorzug der Rotationsfähigkeit der gespeicherten Situationen und damit eine Programmflußsteuerung nach strategischen Gesichtspunkten anstelle "blinder" Lösungssuche auf.

## 2. TRY SYSTEM

Der klassischen FORTH-Programmverzweigungsstruktur

*(flag) IF ELSE THEN bzw.: (flag) IF THEN*

wird eine zweite, "trackende" beiseite gestellt, die äußerlich annähernd äquivalent zu handhaben ist:

*TRY ELSE THEN bzw.: TRY THEN*

Während TRY durchlaufen wird, wird die momentane Programmsitua-

tion auf dem Trackstack abgelegt und der erste Zweig durchlaufen. Derartige Trackstackablagen stapeln sich. Wird an beliebiger Stelle ein Widerspruch erkannt (Kommando *FAIL*), so erfolgt die Übernahme der obersten gespeicherten Situation, und die Programmabarbeitung wird am entsprechenden *ELSE*-Zweig wiederaufgenommen.
Der Umfang der abgelegten Daten erweist sich in der praktischen Anwendung als gering, so daß ein hauptspeicherinterner Trackstack selbst bei kleinen Rechnersystemen (einschl. Einchiprechner) möglich ist. Speichersplittung vermeidend, legt die NILES-FORTH-Implementierung den Datenstack verschieblich auf dem Trackstack an.
Das *TRY*-Grundsystem erfordert auf dem gut ausgebauten FORTH-83-System einen äußerst geringen Aufwand (ca 400 bytes) und erlaubt neben einfachen Anwendungen als Mittel zur Entscheidungsfindung bereits eine regelbasierte Programmierung, eine Anwendung als Inferenzmaschine und dialoggestützte Wissensbearbeitung.
Die Leistungsstärke und Effektivität angesichts des extrem niedrigen Grundaufwands ergibt sich durch die Zugänglichkeit und Erweiterbarkeit FORTH-eigener Mittel /2/,/4/ (einschl. Interpreter, Compiler, Debugger), die Nutzbarkeit eines hochausgebauten Entwicklungssystems /3/ sowie durch die Kombinierbarkeit der *TRY*-Struktur mit den vorhandenen algorithmischen Strukturen.

## 3. USER DATES

User Dates sind Daten, die wichtige Informationen über die Programmsituation enthalten- (Variableninhalte, Pointer, Buffer, Listen u.a.) und deren Sicherung vom Anwender den umweisbaren Funktionen *STORE* und *RESTORE* zugeordnet wird.
User Dates - Zuweisungen können auch dynamisch verändert werden, d.h. je nach Programmphase können unterschiedliche Daten gesichert und im Falle eines Backtrackings in diese Programmphase hinein wiederhergestellt werden.
Es kann zwischen komplett zu speichernden Daten *(Changing Dates)* und solchen mit wachsendem Charakter *(Growing Dates)* gewählt werden. Bei letzteren genügen Pointer, um gespeicherte Zustände wiederherzustellen, die Möglichkeiten der Strategieverarbeitung werden dafür aber eingeschränkt.
Eine besonders interessante Anwendung von Growing Dates stellt die Speicherung des *Dictionary-Pointers* dar. Durch Zuweisungen des Dictionary-Pointers zu der Funktion *STORE* und einer geeigne-

ten *FORGET*-Funktion zu *RESTORE* kann der *TRY*-Mechanismus sehr einfach unmittelbar während der Compilation/Assemblierung eingesetzt werden. Einfache Anwendungen:

- FORTH-Assembler optimiert den Code (z.B.sprungweitenabhängig)
- Compilationsvarianten in Abhängigkeit vom verbleibenden Speicherplatz, von Laufzeiteigenschaften, usw.

## 4. CUT SYSTEM

Für komplexere KI-Aufgaben sind über das Grundsystem hinausgehende Systemmittel erforderlich. Basis dessen ist ein System stapelfähiger lokaler Bereichszuweisungen *(Local Pointer)*. Die Funktion *CUT* "schneidet" alle Alternativen des gültigen lokalen Bereiches ab. Prinziplösung (grob vereinfacht): (*TP* = Trackstack Pointer)

```
VARIABLE  LP       ( Local Pointer )
  {   ( -- )   LP @ TRY  DROP TP @ LP   ELSE LP   FAIL THEN ;
; }   ( -- )   LP @ TRY          @ LP   ELSE LP   FAIL THEN ;
: CUT ( -- )   LP @ TP    ;
```

Beispiel: Die Funktion *?NOT* testet, ob eine Funktion keine Lösung besitzt: `: ?NOT ( cfa -- ) { TRY EXECUTE CUT FAIL THEN DROP } ;`
Die lokalen Bereiche könnten auch automatisch mit den FORTH-Definitionen gleichgesetzt werden (dies entspräche in etwa der Praxis verbreiteter PROLOG-Systeme).

## 5. STRATEGIE-FUNKTIONEN

Strategie ist die Voraussetzung für die Lösung von Aufgaben mit großer oder unendlicher Suchtiefe (z.B. intelligente Programmgenerierung). Die Rotationsfähigkeit des Trackstacks schafft für FORTH die neue Möglichkeit, universell und sehr einfach anwendbare Strategiefunktionen einzubeziehen.
Die Funktion *GRADE* sortiert die bis dahin möglichen Teillösungen des lokalen Bereiches nach einem anwenderspezifischen Bewertungskriterium ein. An dieser Programmstelle erfolgt somit eine partielle Variantenbündelung und Weiterbearbeitung in der Reihenfolge, die durch das Kriterium gegeben ist, d.h. es erfolgt eine schrittweise (bzw. kontinuierliche) *Variantenoptimierung*.
Andere Strategiefunktionen lassen sich ableiten:

```
: COME  ( -- )  TRUE   GRADE ; (einfaches suchen in die Breite)
  MIX   ( -- )  RANDOM GRADE ; (zufallsgesteuerte Variantenausw.)
```

Die Strategiefunktionen wirken insbesondere günstig mit algorithmischen Schleifen zusammen. Prinzip:

*{ BEGIN step value? GRADE ready? UNTIL }*

Von sehr großem Effekt ist dabei die Möglichkeit der Mehrstufigkeit, auch mit unterschiedlichen Bewertungskriterien, z.B.:

*{ { BEGIN step near? GRADE ready? UNTIL } small? GRADE }*

(Aus Laufzeitgründen werden innerhalb der Schleife der Zielstellung nahekommende Situationen bevorzugt, außerhalb wird dann noch einmal nach der Umfänglichkeit der Lösungen sortiert.)

Zentrale Bedeutung hat die Wahl eines geeigneten Bewertungskriteriums. Dieses drückt als ungefähre Tendenz die Qualität des erreichten Arbeitsstandes, d.h. den Äquivalenzgrad von IST und SOLL aus. Bei Schleifen-Strukturen mit kontinuierlicher Optimierung wird die Umfänglichkeit der gefundenen Teillösung (z.B. in Form eines Schleifenindexes) einbezogen.

## 6. METHODEN DES DENKENS

Die Flexibilität und Durchsichtigkeit der Track-Mechanismen erlauben, wichtige Methoden des natürlichen Denkens nachzubilden, um von uneffektiver vollständiger Suche wegzukommen und kompliziertere Aufgaben zu lösen. Äußerst leistungsfähige, vom Anwender erweiterbare Sprachelemente ermöglichen völlig neue Programmiermethoden:

- *Globalentscheidungen* (über den prinzipiellen Lösungsweg, Baustein-Auswahl u.a.), deren Richtigkeit von Zeit zu Zeit, vor allem in wenig erfolgsträchtigen Situationen überprüft und ggf. revidiert werden. Die jeweils gültigen Globalentscheidungskriterien werden durch eine Funktion *GLOBAL-TESTS* geprüft.
  Eine automatische Ursachenanalyse für das Scheitern eines Programmzweiges ergibt sich durch Anwendung eines modifizierten *FAIL*-Kommandos: *: FAIL-WHY ( -- ) GLOBAL-TESTS FAIL ;*
- *Direktiven* für die weitere Abarbeitung, die durch eine Funktion *DIRECTIVES-TESTS* auf ihre Einhaltung geprüft werden.
- *Limitierungen* nach Vezweigungsbreite, Speicherbedarf, benötigter Rechenzeit usw. (Anwendungen z.B. im Sinne von: "Suche 1 Minute nach Lösungen und wähle die zwei besten aus!").
- *Parallelbearbeitung* mehrerer Aufgaben unter Verwendung der *GRADE*-Funktion bezogen auf Prioritäten (für Expertensysteme: komplizierte Aufgaben im Hintergrund langfristig lösen).

- *Schnelle Variantenvorauswahl* mittels Assoziativ-Funktionen, die aus einer zur mathematischen Codierung umgesetzten Situationsbeschreibung die Variantenauswahl steuern. Dadurch kann die Anwendung des relativ zeitaufwendigen Ausprobierens mit Backtracking auf sein natürliches Maß reduziert werden.
- *Echte quantitative Variantenoptimierung* durch universelle Anwendung klassischer Optimierungsverfahren auf verzweigte Funktionen. *MIX* ermöglicht das Finden lokaler Maxima, die mittels *GRADE* eingeordnet werden. Dadurch ist die Bewältigung extrem großer Variantenvielfalt möglich.

## 7. NOTE & FOLLOW

Das Note and Follow - Konzept stellt eine sehr effektive Spezialstrategie dar und besteht darin, den gültigen (d.h. zum Erfolg führenden) Programmweg zu notieren *(NOTE)*, sozúsagen über die durchlaufenen Programmverzweigungen und ausgeführten Backtrackings buchzuführen, um dann erneut (ggf. mehrfach) diesem notierten Weg durch das Programm backtrack- und verzweigungsfrei zu folgen *(FOLLOW)*. Anwendungen:

- *Programm-Simulation* unter Ausschluß zeitaufwendiger oder nicht backtrackfähiger Vorgänge, ggf. unter Einbeziehung echter Simulation physikalischer Vorgänge.
  Sehr zweckmäßig erscheint der unmittelbare Einsatz zur Maschinen- und Robotersteuerung, indem das gleiche Programm zur Simulation und zur direkten Steuerung benutzt wird. Es erfolgt nur aufgabenspezifisch eine Umweisung der Ein-/Ausgabefunktionen:
  Prinzip: *>simulation NOTE ... program ... >process FOLLOW*
  Einfache Anwendungen ergeben sich z.B., indem variable Programme den Bildschirmaufbau inhaltlich und gestalterisch mittels des *TRY*-Mechanismus anpassen und abschließend ausgeben.
- *WHY-Funktion*: die für Expertensysteme notwendige, aber programmtechnisch oftmals sehr aufwendige Entscheidungsbegründung auf Nutzeranfrage läßt sich mittels *FOLLOW* sehr einfach realisieren: *: WHY (S -- ) >printing-why FOLLOW >running ;*

Das *NOTE and FOLLOW* - Konzept bietet auch die Lösung zur Anwendung aller Strategiefunktionen auf Growing Dates, indem vom Programm nicht benötigte Ergebnisse nur abschließend ausgegeben werden, oder indem anstelle einer Programmsituation der *NOTE*-Weg abgespeichert und diese dann mittels *FOLLOW* wiederhergestellt wird.

## 8. DISKUSSION

Durch das FORTH-Backtrack-Konzept ist die vollständige Symbiose zweier im allgemeinen als unvereinbar angesehener Programmiermethoden in einem gemeinsamen Sprachkonzept gegeben.
Die Möglichkeit der Einbeziehung leistungsstarker universeller KI-Mechachanismen beweist aufs neue die unübertroffene Flexibilität und Mächtigkeit des FORTH-Konzeptes /4/.
Die genannten Eigenschaften des *TRY*-Systems machen für FORTH die Arbeit mit KI-Shells, die eine eigene (i.allg. logikorientierte) Sprache besitzen, überflüssig. Falls für spezielle Aufgaben eine solchen Shells ähnliche Nutzeroberfläche erforderlich ist, so ist es zweckmäßig, diese mit den FORTH-typischen Mitteln (als ein Vocabulary) auf Basis des *TRY*-Systems zu realisieren.
Insbesondere kann auf Kopplungen zwischen systemnahen und deklarativen Sprachen verzichtet werden. Viele anstehende Aufgaben aus dem Bereich der Weiterentwicklung der Softwaretechnologie sind überhaupt erst durch ein derartiges Konzept mit vertretbarem Aufwand lösbar. Dies betrifft z.B. Aufgaben automatisierter Implementierung /3/,/5/.
Es ist dabei möglich, über die Grenzen standarter PROLOG-Programmierung hinausgehende Aufgaben (insbesondere von sehr großer Suchtiefe) zu lösen, ohne den ungleich viel höheren Aufwand aufgabenspezifischer Intelligenzlösungen ohne Systemunterstützung (z.B. Schachprogramme) treiben zu müssen.

## Literatur:


/1/ FORTH-83 Standard. Mountain View Presse (August 1983)

/2/ Laxen, H.; Perry, M.: FORTH-83 Model.

/3/ Karadshow,L.: Effektive Mikrorechner-Softwaretechnologie auf FORTH-Basis: Ein hochentwickeltes Programmiersystem. Computer-und Mikroprozessortechnik '88, Fachtagung, Berlin.

/4/ Karadshow; Noack; Wyschowski: Softwareentwicklung mit FORTH. Manuskript, "Mikropozessortechnik" (1988), Berlin.

/5/ Karadshow,L.; Noack, K.: FORTH-Technologie und "künstliche Intelligenz" bei der Entwicklung von Software. FORTH, Fachtagung, Suhl (1988).

/6/ Brodie,L.: Thinking FORTH. Prentice-Hall, Inc.

# MODULA-2: PROGRAMMIERSPRACHE, WERKZEUGE UND PROGRAMMIERBIBLIOTHEKEN ZUR ERSTELLUNG PORTABLER AUTOMATISIERUNGSSOFTWARE AUF DDR-RECHNERN

Hauptmann, St., Reinig, G., Steinke, M., Weber, R.
KdT, Dresden, DDR

## 1. Motivation

Einer der Hauptwege zur Erzielung volkswirtschaftlicher Effekte durch die Mikroelektronik ist die breite Anwendung der Rechentechnik zur Automatisierung in den verschiedensten Bereichen. Den quantitativ und qualitativ steigenden Anwendungswünschen und der rapiden Entwicklung der rechentechnischen Möglichkeiten steht (international und national) ein unzureichendes Potential an Softwareentwicklungskapazität gegenüber. Da andererseits bei größeren Automatisierungsvorhaben auf die Software 50 bis 80 % der Projektkosten entfallen (und davon wieder 50 bis 90 % auf Projektpflege (Fehlerbeseitigung, Weiterentwicklung)), ist die produktivere Erstellung besserer Software ein aktuelles Thema. Wie im /1, 2/ ausführlicher dargestellt, ist dabei die Orientierung auf Softwaremehrfachverwendung (Reuse) eine fruchtbare Richtung. Aus der Sicht eines Bereiches, der Fertigungs-, Prozeß- und Laborautomatisierung betreibt und einen hohen Anteil von "Durchlaufkadern" beschäftigt, ist die Mehrfachverwendung von Softwaremoduln besonders interessant. Eine solche erfordert jedoch

- die Gewährleistung der Rechnerunabhängigkeit (Portabilität) der zu erstellenden Moduln,
- die Verwendung einer Programmiersprache, die wesentlichen Kriterien moderner Softwaretechnologie erfüllt (u.a. Modulkonzept, hohe Sicherheit gegen Programmierfehler, gute Lesbarkeit und Erlernbarkeit ...) und die im Falle der Automatisierungssoftware außerdem Möglichkeiten zur hardwarenahen und Echtzeitprogrammierung besitzt,
- die Bereitstellung von vergleichbaren Entwicklungsumgebungen für unterschiedliche Rechner.

Diese Überlegungen führten zu einer Orientierung auf die Programmiersprache Modula-2 (/2, 3/); im folgenden werden einige Aspekte ihrer Anwendung beschrieben.

## 2. Die neuen Konzepte von Modula-2

Ohne die Sprache Modula-2 hier detailliert vorstellen zu können, soll jedoch ihre direkte Ableitung aus der Vorgängersprache Pascal durch N. Wirth herausgestellt (Beibehaltung solch wichtiger Merkmale wie strenge Datentypbindung, freie Datenstrukturierungsmöglichkeiten, Block-/Prozedurkonzept, go to-freie Programmablaufsteuerungsstrukturen) sowie auf einige neue Eigenschaften verwiesen werden.

### 2.1. Das Modulkonzept

Wie in anderen modernen Sprachen (Ada, Mesa, CHILL) ist die Modularisierung eine zusätzliche Form der Programmstrukturierung.

Zu Moduln "verpackt" werden können z.B.

- Sammlungen von zusammengehörigen Prozeduren (mathematische Standardfunktionen, Konvertierungsroutinen ...),
- Bereitstellung von Datentypen (Implementierung unsichtbar) und der darüber möglichen Operationen (Datenabstraktion, z.B. für die Listenverwaltung, der Zugriff zu E/A-Geräten oder für automatisierungsspezifische Datenstrukturen wie MSR-Stellen ...),
- Zusammenfassung programmweit genutzter Definitionen (z.B. ASCII-Zeichensatz oder Fehlerkodes).

Alle Leistungen, die ein Modul zur Verfügung stellt, werden in der Exportschnittstelle, dem Definitionsmodul, beschrieben, der völlig separat von der Implementierung behandelt werden kann. Das Modulkonzept stellt damit auch programmiersprachliche Mittel zur (Modul-) Bibliotheksbildung bereit und unterstützt die arbeitsteilige Erstellung größerer Programmsysteme.

### 2.2. Zur Portabilität

Das Modulkonzept gestattet die effektive Implementierung einer Hierarchie virtueller Maschinen. Zur Portierung von Software wird eine (möglichst tief liegende, d.h. hardwarenahe) virtuelle Maschine definiert, die auf allen zu versorgenden Rechnern realisierbar ist. Alle Programmteile, die auf diese virtuelle Maschine aufsetzen, sind portabel; die Teile zur Realisierung dieser virtuellen Maschine müssen rechnerspezifisch implementiert werden.

### 2.3. Zur maschinennahen Programmierung

Die Systemprogrammierung (und bestimmte Automatisierungsprogramme) erfordern Zugriffe über Hauptspeicheradressen, E/A-Ports, Betriebssystemrufe, ggf. die Einfügung von Maschinenkode. Diese Möglichkeiten bietet Modula-2 mit dem (rechnerspezifisch implementierten, aber einheitlich definierten) Modul SYSTEM.

### 2.4. Das Koroutinenkonzept

Aufbauend auf den drei elementaren Sprachkonstruktionen zur Rechenprozeßvereinbarung (NEW PROCESS), zum programmgesteuerten und zum interruptgesteuerten Wechsel zwischen Rechenprozessen (TRANSFER und IOTRANSFER) können Moduln zur Synchronisation und Kommunikation bis hin zu Echtzeitbetriebssystemkernen aufgebaut werden.

### 2.5. Offene Feldgrenzen für Prozedurparameter

Modula-2 ermöglicht Felder beliebiger Länge als Parameter. Zur Ermittlung der tatsächlichen Länge innerhalb der Prozedur existiert die Standardfunktion HIGH.

## 3. Modula-2-Programmiersysteme

Modula-2 ist eine vergleichsweise junge Programmiersprache. Das ihr inherente Portabilitätspotential unterstützte jedoch die schnelle Verbreitung von Programmiersystemen, so daß u.a. ab A7100 für alle Robotron-Rechner vom Hersteller Modula-2-Programmiersysteme vertrieben werden. Die z.Zt. in der DDR benutzten Programmierumgebungen (Tabelle 1) unterscheiden sich bezüglich Umfang und Komfort erheblich; einzelne Spitzencharakteristika (z.B. Komfort des Turbopasal V.4 oder Effektivität bestimmter FORTRAN-Compiler) werden noch nicht erreicht; jedoch wird jeweils die komplette Sprache mit all ihren softwaretechnologischen Vorteilen unterstützt, so daß im Zusammenhang mit einheitlichen Stndardbibliotheken volle Portabilität vom 8bit-BC bzw. -PC bis zum K1840 bzw. zu ESER und eingeschränkte Portabilität bis hinab zu 8-bit-Einkartenrechnern ermöglicht wird. (Für die 8-bit-Systeme existiert in der DDR kein Kodegenerator, die realisierte Zwischenkodeinterpretierung bringt zwar hohe Kodekompaktheit aber auch erhebliche Laufzeitverlängerungen).

## 4. Beispiele für die Arbeit mit Modula-2-Systemen

Die Anwendungsbreite der "Modula-2"-Softwaretechnologie verdeutlichen folgende teils abgeschlossene, teils in Bearbeitung befindliche Vorhaben, die unter maximaler Nutzung obiger Reuse-Strategie bewältigt wurden/werden.

- Portable Bibliothek für allgemeine Aufgaben (alphanumerische, window-orientierte Terminal-Ein/Ausgabe, pixelorientierte Grafik, universelles Filesystem, Konvertierungen, Zeichen-/Stringbearbeitung, Kommandozeilen-verarbeitung, Fehlerbehandlung, Überlagerungslader, mathematische Standardfunktionen und Matrixgrundoperationen, Zeitdienste, Testhilfskomponenten);
- Portable Bibliothek für regelungstechnische Entwurfsaufgaben (Matrixoperationen, Polynome, komplexe Arithmetik, Gleichungssysteme, Nullstellen, ausgewählte nichtlineare Optimierungsverfahren, Transformationen regelungstechnischer Modelle);
- Entwicklungssysteme zur Modula-2-Softwareerstellung für 8bit-Einbaurechner oder 8bit-Kleinrechner auf CP/M-, UDOS- oder MS-DOS-(DCP-)Rechnern (z.B. zur Programmversorgung für Frontend-Rechner in der Labor-/Prüfstandsautomatisierung);
- Weitgehend portabler (modularer) Echtzeitbetriebssystemkern XMOD (zunächst für 16bit-PC) mit den wesentlichen Merkmalen moderner Echtzeit-Betriebssysteme und ausgefeilter Fehlerbehandlung.

## 5. Schlußbemerkung

Die bisherigen Erfahrungen mit Modula-2, die ganz wesentlich durch die Vorleistungen bzw. die freundliche Unterstützung der Fachkollegen aus der HUB, von RPD und aus dem IIR der AdW gefördert wurden, gestatten die Schlußfolgerung, daß mit dieser Sprache in der DDR eine softwaretechnologische Basis für die effektive(re) Erstellung komplexer (Automatisierungs-) Softwareprodukte vorhanden ist.

Tab. 1 Einige in der DDR bekannte MODULA-2-Systeme

| Rechner-klasse | Betriebs-system | Programm-system | Prinzip | Programmierumgebung/ Besonderheiten |
|---|---|---|---|---|
| ESER | ESER - PSU | ESER-M2 (HUB, TUK) | Compiler | |
| K1840 | SVP1800 MUTOB1800 | SVP-M2 MUTOB-M2 (Robotron) | Compiler | Objektfilevertraeglichk. mit C- und Fortran; Test mit symbol. Debugger des Betriebssystems |
| A7150 EC1834 | DCP | DCP-M2 (Robotron) | Compiler Linker | Bibliotheksunterstuetz. 2 Speicherkonzepte |
| IBM-Kompat. | MS-DOS | LOGITECH-MODULA-2 | integriertes System | syntaxunterstuetz. Editor; Exe-File-Erzeugung, Laufzeit- und post-mortem-Debugger, Dekoder |
| | | M2 - BDS (ITC) | integriertes System | syntaxgesteuerter Editor Bibliotheksverwaltung |
| | | JPI - M2 Taylor-M2 | integriertes System integriertes System | residenter Compiler; bezueglich Geschwindigkeit und Nutzerober-flaeche vergleichbar mit TURBO-PASCAL |
| | | INDIA (AdW IIR) | PALM-Compiler | komplettes Programmiersystem Laufzeitdebugger |
| A7100 | SCP1700 | SCP-M2 (Robotron) | Compiler, Linker | analog zu DCP-M2 |
| SKR K1600 | RSX11 MOOS 1600 | RSX-M2 (TUD 89) | Compiler, Linker | Taskerzeugung, post-mortem-Debugger, Dekoder |
| 8-Bit-PC PC1715 A5120 | SCP, CPA UDOS | MODULA-M2I (TUD 89, HUB) | Compiler, Linker, M-Code-Interpreter | Interpreterlaufzeitsystem Dekoder, CMD-File- bzw. Prozedurfileerzeugung |
| | CP/M | Turbo-M2 (Borland) | integriertes System | |

## 6. Literatur


/1/ C.V. Ramamoorthy u.a.: Programming in the Large. IEEE Trans. on Software Engineering. vol SE-12 (1986) NO. 7, S. 769 - 783

/2/ D. Balzer, G. Reinig, St. Hauptmann: Softwarewiederverwendung - ein notwendiges Grundprinzip effektiver Softwaretechnologie. messen.steuern.regeln, 30 (1987) H. 2, S. 60 - 63

/3/ N. Wirth: Programmierung in Modula-2. Berlin/Heidelberg/New York/Tokyo: Springer Verlag 1987

/4/ U. Lindner, R. Trautloft: Grundlagen der problemorientierten Programmentwicklung Verlag Technik, Berlin 1987


In /2/ sind weitere (internationale) Literaturquellen zur Softwaretechnologie angegeben.

# Multitask-Programmabarbeitung im Betriebssystem CP/M-86

Jorke, G. Rostock/DDR

Echtzeitbetriebssysteme, wie z.B. BOS1810 und RMX86, besitzen eigene Entwicklungswerkzeuge für die Programmentwicklung von Assembler- und Hochsprachprogrammen. Diese beruhen auf einer eigenen Dateiorganisation mit speziellen dafür notwendige Dienstprogrammen. Die Dokumentationen eines Echtzeitbetriebssystems umfaszt viele 1000 Seiten.
Die sich daraus ergebenden Anforderungen an den Nutzer haben die Anwendung von Echtzeitbetriebssystemen stark behindert.
Diese Tatsache hat zu alternativen Lösungen geführt. Auf der einen Seite wird versucht, Standardbetriebssysteme (z.B. MS-DOS) mit Eigenschaften von Echtzeitbetriebssystemen, insbesondere der Multitaskabarbeitung, zu ergänzen. Zum anderen wird in weiterentwickelte Echtzeitbetriebssysteme die Programmentwicklungstechnik von Standardbetriebssystemen eingeschlossen (z.B. RMXII.3 für das AT-kompatible Rechnersystem 120 von Intel).

Im folgenden wird die Multitaskerweiterung rMTS für das Betriebssystem CP/M-86 bzw. SCP1700 vorgestellt.
Das Multi-Task-System rMTS dient der Entwicklung von Multitaskprogrammen, die die Subsysteme NUCLEUS und SYSTEMDEBUGGER des Echtzeitbetriebssystems BOS1810 (kompatibel zu RMX86) einschlieszen, mit der Programmentwicklungstechnik des Betriebssystem SCP1700 (kompatibel zu CP/M-86).
Die Programmentwicklung mit dem Einschlusz der NUCLEUS-Systemrufe kann in Assemblersprache oder in der Programmiersprache C erfolgen.
Die Multitaskprogramme können mit gewissen Einschränkungen die Treiberfunktionen (BDOS-Funktionen) des Betriebssystems SCP1700 bzw. CP/M-86 benutzen.

Die charakteristischen Eigenschaften echtzeitfähiger Programme liegen in der quasiparallelen Abarbeitung einzelner Programmteile, der Tasks und der Unterstützung der Interruptverarbeitung.
Die Formulierung eines Anwenderprogramms aus verschiedenen Tasks für die Behandlung unterschiedlicher Ereignisse in einem Prozesz führt zu einer weitgehenden Entflechtung von Teilprogrammen und zu einer hohen Flexibilität bei der Programmentwicklung und bei Aenderungen.

Ein Multitaskanwenderprogramm übernimmt Bestandteile des Betriebssystems für die Organisation der Multitaskverarbeitung.
In einem Multitaskprogramm wird die Abarbeitung der Tasks von einem Task-Scheduler innerhalb des NUCLEUS nach dem Prioritätsprinzip gesteuert.
Tasks werden vom NUCLEUS als Objekte verwaltet. Weitere Objekttypen dienen der Resourcenverwaltung des Rechners (Job), der dynamischen Speicherzuweisung (Segment), dem Informationsaustausch zwischen den Tasks (Mailbox), der Synchronisation der Taskabarbeitung (Semaphore und Region).

Jede Task erhält einen Prioritätswert, einen der Ausführungszustände: 'rechenbereit' (ready), 'schlafend' (asleep) oder 'suspendiert' (suspended), einen eigenen Stackbereich und meist einen eigenen Datenbereich.
Die jeweils höchst priorisierte, rechenbereite Task wird abgearbeitet.
In den Wartezeiten der höchstpriorisierten Task übernimmt die nächst höher priorisierte, rechenbereite Task den Prozessor.
Wartezeiten können prozeszbedingt (z.B. bei E/A-Operationen) auftreten oder durch eine Vielzahl von Systemrufen ausgelöst werden, so dasz auch niedrig priorisierte Tasks in die Abarbeitung kommen.
Der Objekttyp Job umfaszt ein Kontingent an Rechnerresourcen und Objekten. Durch die Bildung verschiedener Jobs kann der Arbeitsbereich des Rechners in voneinander abgegrenzte Bereiche aufgeteilt werden. Eine Task kann innerhalb eines Jobs dessen Resourcen auf weitere Jobs ("Kinderjobs") aufteilen.
Die Gesamtresourcen des Rechners werden von einem Wurzeljob (rootjob) verwaltet, der vom NUCLEUS erzeugt wird und dessen Initialtask (Wurzeltask) nach dem Anlauf des System gestartet wird.
Innerhalb der Wurzeltask werden ein oder mehrere Anwenderjobs gebildet, die die Anwendertasks enthalten.

Die Reaktionsfähigkeit eines Echtzeitprogrammes auf externe Ereignisse wird durch die Bildung von Interrupttasks gewährleistet. Eine Interrupttask wird durch eine Interruptanforderung aus der Hardware in den rechenbereiten Ausführungszustand überführt.

Multitaskassemblerprogramme können im Betriebssystem CP/M-86 (SCP1700) mit den Assemblerprogrammen ASM86 oder RASM86 entwickelt werden.
Eine Interfacebibliothek vermittelt den Anschlusz an die NUCLEUS-Systemrufe.

Wesentlich komfortabler als die Programmierung in Assembler ist die Nutzung von Hochsprachen. Für die Multitaskerweiterung rMTS wurde die Möglichkeit geschaffen, die Programmiersprache "C" zur Programmierung zu verwenden.

Die BDOS-Funktionen des Betriebssystems CP/M-86 bzw. SCP1700 können, bis auf einige Ausnahmen, in Anwendertasks aufgerufen werden. Dabei ergeben sich die folgenden Einschränkungen:

a) Eine BDOS-Funktion darf nicht durch einen Zeitgeberinterrupt des MTS-Systems unterbrochen werden.
   Die im System rMTS über den Softwareinterrupt INT224 ansprechbaren BDOS-Funktionen werden durch ein Rahmenprogramm gegen Zeitgeberinterrupts geschützt.
b) Da die BDOS-Funktionen nicht wiedereintrittsfähig sind, ist ein Taskwechsel innerhalb des Aufrufs einer BDOS-Funktion nicht möglich, wenn die die Abarbeitung übernehmende Task die unterbrochene BDOS-Funktion aufruft. Durch das Sperren

externer Interrupts (z.B. BREAK-Taste) während der Abarbeitung der BDOS-Funktion wird ein Taskwechsel vermieden.

c) Ein verschachtelter Zugriff mehrerer Tasks auf die Diskettenlaufwerke ist nicht möglich.

d) Mit den oben genannten Einschränkungen müssen Konsolzugriffe zeichenweise erfolgen. Um zusammenhängende Zeichenfolgen für die Ein- und Ausgabe zu und von einer Task zu sichern, kann der Konsolzugriff über ein Semaphore im Anwenderprogramm verwaltet werden.

e) Die BDOS-Funktion 0 hat im System rMTS das Löschen der aufrufenden Task und das Verbleiben im System rMTS zur Folge.

Die Zugriffsverwaltung für Ein- und Ausgabeoperationen der Konsole kann der Konsolhandler des rMTS-Systems übernehmen. Der Konsolhandler wird mit dem Applicationloader geladen und als Task kreiert. Die Kopftask des Applicationloaders erzeugt zwei weitere Tasks mit hoher Priorität für die Verwaltung der Konsoleingabe und der Konsolausgabe.
Ausgaben auf die Konsole können von allen Anwendertasks über SEND_MESSAGE-Systemrufe an die Konsolausgabetask organisiert werden.
Tastatureingaben werden durch die Anwendertasks von einer Konsoleingabetask angefordert. Eine Anwendertask musz ein Eingabesegment an die Empfangsmailbox der Konsoleingabetask senden.

Die Entwicklung von Multitaskprogrammen mit der Entwicklungstechnik des Betriebssystems CP/M-86 bzw. SCP1700 erfordert ein Hilfsmittel, mit dem getrennt entwickelte Programmteile als Tasks an den Taskscheduler des NUCLEUS angeschlossen werden.
Der modulare Charakter der Multitaskabarbeitung mit dem quasiparallelen Betrieb vieler Tasks erfordert, zumindest im Programmentwicklungsstadium, eine flexible Zusammenstellung von Multitaskprogrammen.
Deshalb wird nach dem Start des Multitasksystems rMTS innerhalb eines First-Level-Jobs der Applicationloader als Task aktiv. Mit dem Applicationloader können beliebig viele weitere Tasks erzeugt werden.
Der Applicationloader übernimmt die Ladefunktion für die unter CP/M-86 bzw. SCP1700 erzeugten Programmdateitypen und die Bildung von Tasks. Als Dateiformen werden -.CMD- und -.H86-Dateien berücksichtigt.
Die geladenen Programmdateien werden anschlieszend als Tasks kreiert. Die dafür notwendigen Parameter werden durch die Kopfparameter der Datei, durch dynamische Speicherzuweisungen des NUCLEUS und im Bediendialog festgelegt.

Um Multitaskprogramme erfolgreich testen zu können, enthält das System rMTS zwei leistungsfähige Werkzeuge:
Der A7100-Monitor ermöglicht das Abarbeiten von Programmen und Programmteilen im Einzelschrittbetrieb, das Anzeigen und Ändern von Speicherbereichen und Registern.
Die Leistungen des Systemdebuggers sind rMTS-spezifisch und ermöglichen die Anzeige und die Interpretation von rMTS-Objekten.

# 16-BIT-ECHTZEIT-MULTIRECHNERSYSTEME MIT E/A-PROZESSOREN 8089

Stachowiak, T., Rostock/DDR

Eine der wichtigsten Methoden der Effektivitätserhöhung der Computersysteme ist die Steigerung des Parallelitätsgrades der auszuführenden Operationen. Dies kann erreicht werden durch Systeme, die sich aus mehreren zentralen, parallel arbeitenden Rechnern zusammensetzen. Außer der Erhöhung der Leistungsfähigkeit besitzen parallele Systeme weitere potentielle Vorteile:

- -Erhöhung der Zuverlässigkeit, indem eine Aufgabe durch mehrere Rechner abgearbeitet wird und die Ergebnisse periodisch verglichen werden.
- -Möglichkeiten der Modularität eines Systems hinsichtlich der Modifizierung und Weiterentwicklung.

Die Entwicklung von Mikroprozessoren der 16-bit-Generation hat die technischen und ökonomischen Voraussetzungen für den breiten Einsatz von Multirechnersystemen geschaffen.

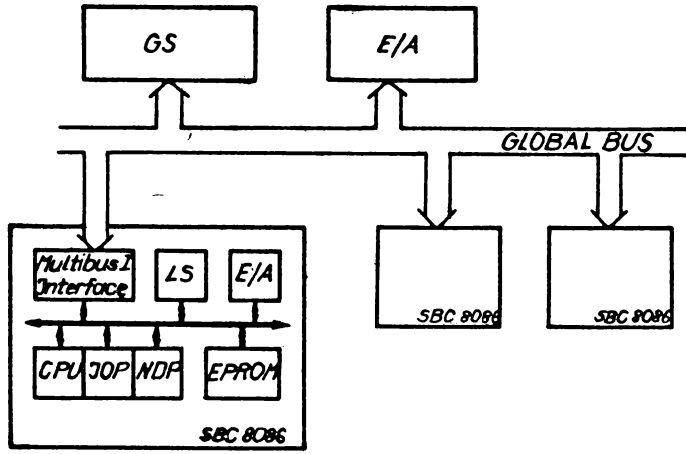
Werden an einen gemeinsamen (lokalen) Bus mehrere Mikroprozessoren angeschlossen, so entsteht ein **Multiprozessorsystem**, bei dem alle Prozessoren Zugriff zum gesamten Speicher und zu allen E/A-Modulen haben. Spezielle immer mehr an Bedeutung gewinnenden Multiprozessorsysteme zeichnen sich durch den Einsatz einer universellen CPU und mehreren spezialisierten Coprozessoren für arithmetische oder E/A-Operationen aus.

Werden Ein- oder Mehrprozessorsysteme an einen globalen Bus angeschlossen, so entsteht ein **Multirechnersystem.** An diesem Bus befinden sich der globale Speicher und gegebenfalls gemeinsame Schnittstellen für E/A-Geräte.

Die weiteren Ausführungen beziehen sich auf das an der WPU Rostock realisierte Multirechnersystem mit folgenden Charakteristika:

Das Multirechnersystem besteht aus drei 16 Bit Single-Board-Computern auf Basis des Mikroprozessors 8086 mit:

- -0,5MByte dRAM
- -16KByte EPROM
- -Coprozessoren 8087, 8089
- -Multirechnerfähigkeit durch den Einsatz des Multibus I Arbiters 8289.




Architektur des realisierten Multirechnersystems

Das System besitzt einen globalen Speicher von der Größe 0,5MByte, über den die Kommunikation zwischen den SBC's vorgenommen wird.

Basis für die Implementierung der Prozeßsoftware ist auch bei Multirechnersystemen ein geeignetes Echtzeitbetriebssystem. Es muß alle Forderungen erfüllen, die auch an ein Einprozessorsystem gestellt werden. Insbesondere muß eine geeignete Kommunikation und Synchronisation der globalen Zugriffe gewährleistet werden. Das Echtzeitbetriebssystem läuft nicht auf einem zentralen Prozessor ab, sondern ist im gesamten Multirechnersystem verteilt und mehrfach vorhanden.

Im realisierten System wurde auf jedem der SBC's der rMTS-Echtzeitkern, kompatibel zu BOS 1810- und RMX86-Kern, implementiert. Die Kommunikation zwischen den Echtzeitkernen wird über logische Schnittstellen, genannt Ports, abgewickelt. Die physische Datenübertragung übernimmt der sich auf jedem SBC befindende Coprozessor 8089. Außerdem erfüllt der IOP noch zwei weitere Aufgaben:

-Abfrage von globalen Systemvariablen (Semaphore)
-Durchsuchen der global angelegten Objektverzeichnisse nach einem Objektnamen.

Der E/A-Coprozessor ist ein selbständiger 16-bit-Mikroprozessor, der auf einem Chip die Merkmale einer CPU und eines DMA-schaltkreises vereint. Der 8089 bildet mit der 8086-CPU ein eng gekoppeltes Multiprozessorsystem. Die Kommunikation der beiden Prozessoren untereinander erfolgt durch den lokalen RAM. In dem im Vortrag vorgestellten Multirechnersystem sollen folgende spezielle Eigenschaften des E/A-Prozessors genutzt werden:

-bus load limit

-direct memory acces

In der Betriebsart "bus load limit" führt der E/A-Prozessor sein im Speicher oder E/A-Bereich abgespeichertes Programm, indem alle 128 Systemtaktzyklen der CPU ein Befehl automatisch ausgeführt wird. Da die 8086-CPU durch ihren Befehlswarteschlangenmechanismus nicht immer den Bus benötigt um Befehle abzuarbeiten, führt diese Betriebsart zu weitestgehender Parallelisierung der Arbeit der beiden Prozessoren. Konkret heißt es, daß o.g. Aufgaben des E/A-Prozessors im realisierten System transparent zu den zentralen CPU's erfüllt werden können.

EMR-ENTWICKLUNGSSYSTEM UNTER SCPX (CPA)

Reichl, P., Dr.-Ing., KDT Ilmenau, DDR

## 1. Einleitung

Zur Programmentwicklung, Fehlersuche und Programmtestung werden heute weitgehend Mikrorechnerentwicklungssysteme genutzt. Um auch für die Einchipmikrorechner der Serie U88xx vorhandene Entwicklungssysteme mit ihren vielfältigen Möglichkeiten auszunutzen, wurde eine Emulatorkarte /1/ im K1520-Format und ein Betriebssystem - auch vielfach als Monitor bezeichnet - entwickelt. Dieser Monitor, der auf bestimmte Kommandowörter reagiert, dient zur Steuerung der Emulaterkarte und läuft zur Zeit auf den Entwicklungssystemen MC 80/20, MRES (MEOS 1521) und unter SCPX (CPA) auf K8915, BC A5120/30 sowie PC 1715.
Der Anschluß der Anwenderlogik erfolgt über Adapterkabel (vgl. dazu /2/ und /3/), so daß die Fehlersuche mit angeschlossenem EMR-Objekt in der Hard- wie Software vorgenommen werden kann.

## 2. Schaltungskonzeption

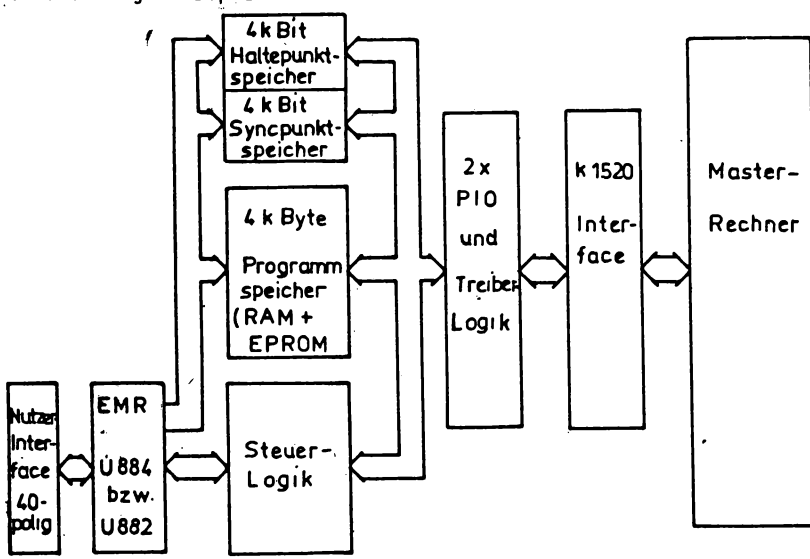



Bild 1: Blockschaltbild des EMR-Emulators

Die Schaltung des Emulators (Bild 1), die als parallele E/A-Schnittstelle zum Entwicklungssystem (Masterrechner) konzipiert wurde, läßt sich in fünf logisch zusammengehörende Funktionsgrup-

pen zusammenfassen.

- E/A-Ports

  Die gesamte Steuerung des Emulators erfolgt über 32 E/A-Leitungen. Damit stellt die Emulatorkarte für den Masterrechner lediglich eine E/A-Karte (ohne INT-Betrieb) dar. Die E/A-Leitungen dienen der

  Steuerung des Emulators vom Masterrechner aus
  Rückmeldung über Ruhe- bzw. Laufzustand des EMR
  Rückmeldung über INT-Anforderungen durch den EMR
  Adressierung und Datenübergabe an den Programm-, Haltepunkt- und Synchronisations(Trigger-)punktspeicher und deren Steuerung.

- Treiberlogik zum Anschluß an K1520-Systeme

  Die Emulatorkarte hat als E/A-Gerät völlig frei wählbare E/A-Adressen, einen getriebenen Datenbus, RDY-Signalerzeugung sowie IODY-Auswertung

- Speichereinheit

  Auf der Emulatorkarte befindet sich ein Speicherblock, der aus folgenden Teilen besteht:

  | | |
  |---|---|
  | 4 k Byte Programmspeicher RAM | (8 x U 214) |
  | 4 k Byte Programmspeicher EPROM | (2732 oder 2716) |
  | 4 k Bit Haltepunktspeicher RAM | (4 x U 202) |
  | 4 k Bit Synchronisationspunktspeicher | (4 x U 202) |

- Speicherlogik (Adressierungslogik)

  Die Speicherlogik dient zur Auswahl der im Speicherbereich ØØØØ - ØFFFH liegenden 8 Speicherschaltkreise U 214 D

- Logik zur Steuerung des EMR

  Diese Logik beinhaltet

  Logikanschlüsse zur Anwenderhardware (ext. $\overline{\text{RESET}}$, CLEAR, $\overline{\text{CLEAR}}$, Quarztakt)
  Logik zur Koordinierung des Programmspeicherzugriffs (EMR oder Masterrechner)
  Logik zur Steuerung von Echtzeitlauf- und Ruhebetrieb des Einchipmikrorechners

## 3. Betriebssystem (Monitor)

### 3.1. Programmstruktur des Betriebssystems

Die Emulatorsoftware ist logisch in drei Ebenen gegliedert (Bild 2).

In diesen drei Ebenen erfolgt auch die Ausführung eines Kommandos von der Eingabe bis zur Fertigmeldung. Die oberste Pro-

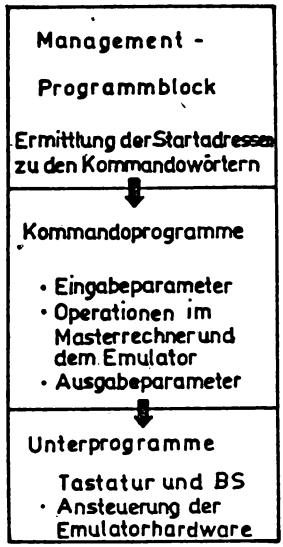



Bild 2: Programmstruktur

grammebene bildet der Management-Programmblock, unter CP/M als CCP bekannt. Von hier aus werden die Startadressen der Kommandoprogramme ermittelt, um dann in die mittlere Ebene zu verzweigen. Ist ein eingegebenes Kommando, das auch über Kürzel eingebbar ist, nicht interpretierbar, weil z. B. flasch eingegeben, verbleibt die Steuerung in der obersten Ebene. Korrekturen oder Neueingaben sind dann erforderlich. Die mittlere Ebene bilden die Kommandoprogramme. Diese Module bestehen aus einer Folge von Aufrufen von Programmen der untersten Ebene. Jeder Modul enthält einen Organisations-, Ausführungs- und Abschlußteil. Die Kommandomodule erfragen Eingabeparameter, führen Operationen im Masterrechner und der Emulatoreinheit aus und übergeben, wenn erforderlich, Ausgabeparameter. Wird dabei ein Fehler festgestellt, gelangt die Steuerung wieder in die oberste Ebene.

Die unterste Ebene ist ein Programmpaket, das von beiden übergeordneten Ebenen genutzt wird. In ihm befinden sich Unterprogramme zur Unterstützung von Tastatur- und Bildschirmarbeit sowie Unterprogramme zur Ansteuerung der Emulatorhardware.

3.2. Leistungsfähigkeit des Betriebssystems unter SCPX

Die Emulatorsoftware wird über Auswahltasten nach dem Systemstart an die jeweilige Masterrechnerhardwarekonfiguration angepaßt und gestattet nach einer Grundinitialisierung das Arbeiten als EMR-Entwicklungssystem über die Kommandos

(C), COMMANDS, (H) HELPLEVEL, (N) NEW,
(RP) RESETPULS, (IE) INIEMR, (T) TEST, (DS) DISK, (PM) PROGMEMORY,
(DO) DOCUMENTS, (CK) CHECK, (PO) POINTS, (PC) PC, (RG) REGISTER,
(O) ONE, (RA) REASSEMBLER, (S) SYSTEM

vollständig im Dialogbetrieb. Diese Kommandos führen meist über

weitere Menüs zu den gewünschten Funktionen.
So kann ausgewählt werden, ob die Programmtestung im Einzelschritt-, automatischem Schritt- bzw. RUN-Betrieb erfolgen soll. Die Testmöglichkeit in den Schrittbetriebsarten ist jedoch auf den 4kByte-Speicherraum begrenzt, auf den die Steuerlogik (Bild 1) ausgelegt ist. Funktionsfähige Programmodule können dann bei weiterem Speicherplatzbedarf auf den extern anschließbaren Speicherraum ausgelegert werden.
In den beiden Schrittbetriebsarten wird sowohl die Belegung der Register der aktuellen Arbeitsregistergruppe als auch der abzuarbeitende Befehl im Maschinencode einschließlich dessen Reassemblierung auf dem Bildschirm angezeigt.
Drüber hinaus informiert die Bildschirmausschrift im Schrittbetrieb über den PC-Stand, die Datenbewegungen über die Portregister, die INT-Anforderungen und die INT-Maskierung sowie über den INT-Status (DI, EI). Mit dem Kommando "Register" wird die Registerbelegung des EMR auf einen Blick sichtbar gemacht.
Load- und Save-Treiber ermöglichen eine bequeme Diskettenarbeit mit dem Emulator bzgl. der über die Assembler SYS8 bzw. XRMAKZ8 (oder ähnlicher) erstellten HEX-Files im INTEL-HEX-CODE bzw. direktem HEX-CODE (Phase).
Das Arbeiten mit Hardware-Haltepunkten im gesamten 4k-Speicherraum wie auch das Setzen von Hardware-Triggerpunkten für die externe x-Ablenkung von Oszillografen oder für die Triggerung von Logikanalysatoren erlaubt eine komfortable Hard- und Softwarefehlersuche bei der EMR-Objektrealisierung. Bei Anschluß eines Druckers kann über die gesamte Emulationssitzung oder auch nur teilweise Protokoll geführt werden.

## 4. Zusammenfassung

Mit dem Betriebssystem V3.10 unter SCPX wurde das von der Technischen Hochschule Ilmenau, Sektion INTET, angebotene Einchipmikrorechner-Entwicklungssystem /1/ zu einem leistungsfähigen, den heutigen Ansprüchen gerecht werdenden Entwicklungswerkzeug für die Familie der Einchipmikrorechner U88xx weiterentwickelt.

## 6. Literatur


/1/ U-884-Emulatorkarte und Steuerprogrammsystem rfe 1986, H. 10; S. 612

/2/ Müller, J.; Thein, M.: Entwicklungs- und Testmodul für Einchip-MR-Programme rfe 1983, H. 11; S. 721-723

/3/ Franke, K.; Leichsenring, A.: Entwicklungsmodul für EMR rfe 1984, H. 1, S. 8-10

# ASPEKTE BEI MULTIPROZESSORENENTWICKLUNGEN

Bernstein, K., Prof. Dr. sc. techn. Dr. sc. oec., Dr. phil.
Leipzig, DDR

Mit der immer umfassenderen Anwendung der informations- bzw. wissensverarbeitenden Systeme in allen gesellschaftlichen Bereichen stiegen und steigen die Anforderungen an diese Systeme. Gemäß solchen wesentlichen anwendungstechnischen Aspekten wie z. B.

- Verarbeitungsgeschwindigkeit
- Systemsteuerung
- Kommunikation
- Fehlertoleranz, Verfügbarkeit
- Preis/Leistungs-Verhältnis

und ihrer qualitativen und quantitativen immer besseren Umsetzung, gewinnen Fragen der Realisierung neuer Rechnerarchitekturen immer mehr an Bedeutung. Das wird durch die Tatsache unterstrichen, daß der Einfluß der Mikroelektronik auf einzelne dieser Aspekte (z. B. Verarbeitungsgeschwindigkeit) in dem Maße verloren geht, wie sich die Parameter der Bauelemente der Mikroelektronik physikalischen Grenzen nähern (z. B. Signalverzögerung gegen Null).

Mit Einschätzungen /1/ /2/, daß z. B. die zeit- und kostenbedingte Notwendigkeit

- technische Entwicklungen (Flugzeugbau, Automobilindustrie)
- biologische Untersuchungen (Modelle der Molekularbiologie)
- wissenschaftliche Theorie-Modelle (Klimaforschung, Meteorologie) vollständig mathematisch zu simulieren an Rechenleistungen bis 10 Tera-FLOPS gebunden sind
- Einprozessorsysteme auf Grund physikalischer Grenzen nicht wesentlich schneller als 10 Giga-FLOPS sein können

wird die Forderung nach neuen Rechner architekturen objektiv untermauert.

Ausgangspunkt der Bewertung bekannter bzw. neuer Rechnerarchitekturen ist neben ihrer Leistungseinschätzung (Benchmark-Test) eine allgemeine Qualitätseinschätzung, die sich an den Begriff der "Rechnerarchitektur" anlehnt und zu einer Klassifikation von Rechnerarchitekturen führt.

Aus den verschiedenen Versuchen der Klassifikation von Rechnern

sind die Klassifizierungen nach
- Flynn /3/
- Händler /4/
- Giloi (Taxonomie) /5/
wesentlich.
Besonders die Taxonomie nach Giloi - nach der sich ein Rechner als eine Relation von R = R (P, O, S) (P = Menge von Zwecken, O = Menge der Operationsprinzipien, S = Menge der Strukturklassen) - ist hervorzuheben. Eine Analyse der heutigen und künftigen Entwicklung zeigt, daß eine Rechnerklassifikation hinsichtlich der Relation R = R (A, V, S)
A = Anwendungsaspekt
V = Verarbeitungselement ( p)
S = Struktur
günstige Eigenschaften aufweist, zumal das Operationsprinzip eines Rechners im allgemeinen nicht hardwarespezifisch ist. Durch die dargestellte Relation als Grundlage einer Rechnerklassifikation wird der insbesondere Einfluß der Leistung des Verarbeitungsprozessors auf die Systemleistung deutlich.

Ausgehend von der Analyse verschiedener Rechnerarchitekturen zeigt sich, daß u. a. Multiprozessorsysteme geeignet sind, eine Vielzahl von anwendungstechnischen Aspekten positiv zu beeinflussen, auch wenn es direkte bzw. indirekte Vorbehalte gegen solche Rechnerarchitekturen gegeben hat und gibt wie z. B.
- Gesetz von Losch
- Minskysche Vermutung
- Komplexitäts-These
- Spezialrechner-These
Diese Vorbehalte können durch moderne, realisierte Multiprozessorsysteme widerlegt werden.

Hinsichtlich einzelner bzw. mehrerer ausgewählter anwendungstechnischer Aspekte ist die Relation R = R (V, S), die in einer Matrix VXS dargestellt werden kann, von besonderer Bedeutung für die Realisierung allgemeiner bzw. spezifischer Multiprozessorsysteme.
Um Multiprozessorsysteme sowohl hinsichtlich ihrer allgemeinen Bewertung (R = R (A, V, S) als auch hinsichtlich ihrer spezifischen Eigenschaften (z. B. R = R (V, S)) verfassen zu können, gibt es eine Vielzahl mathematischer Ansätze.

Wesentliche Ansätze sind u. a.:

- Erlanger Klassifikationsschema (Händler)
- Verbindungsnetzwerk für Multiprozessorsysteme (Bhuyan, Agrawal)
- S-Net-Graphen für SIMD-Computer (Krygiel)
- Beschreibung und Entwurf von Mehrprozessorsystemen (Maisel, Merlin)
- Modelle für Multiprozessorsysteme mit Multibusverbindungsnetzwerken (Madge, Haves, Buzzard, Winsor)

Ausgehend von diesen Ansätzen kommt es darauf an, die dikutierte Relation für Multiprozessorsysteme als mathematisches Modell für Multiprozessorsysteme weiter zu untersetzen.
Hinsichtlich der Bewertung der Verarbeitungselemente und ihrer Nutzung in Multiprozessorsystemen ergeben sich als wesentliche Parameter:

- Verarbeitungsbreite (8-bit, 16-bit, 32-bit)
- Taktfrequenz
- Multiprozessorfähigkeit

Die Analyse bekannter universeller Mikroprozessoren zeigt, daß bei Mikroprozessoren mit einer Verarbeitungsbreite von 16-bit der Intel-Prozessor 80286 sehr gute Eigenschaften zeigt. Das gilt auch für die Mikroprozessor mit 32-bit Datenbreite 80386.

Für spezifische Anwendungsfälle (z. B. Bildverarbeitung, Grafiksysteme) sind auch Signalprozessoren geeignet (z. B. TMS 320). Besonders gute Eigenschaften zur Realisierung von Multiprozessorsystemen zeigen die Transputer von Inmos T 414, F 424, T 800.

Gemäß den unterschiedlichen Anwendungsmöglichkeiten gibt es derzeit eine Vielzahl von Verbindungsstrukturen für Multiprozessorsysteme. Eine Bewertung der einzelnen Möglichkeiten ist unter verschiedenen Gesichtspunkten durchzuführen und entsprechende Optimalkriterien zu erarbeiten.

Literatur:

/1/ Gentzsch, W.: Parallelität in Supecomputern
CHIP PLUS 3 März 1988

/2/ Händler, W.: Topologien, Komplexität und Effizienz von Multiprozessoren
PARS-Mitteilungen 1987

/3/ Flynn, M: Some Computer Organisation and their Effektivness
JEEE Transactions on Computers C-21
(1972) Nr. 9 S. 948 - 960

/4/ Bode, A., Händler, W.: Rechnerarchitektur
Teil 1: Grundlagen und Verfahren
Teil 2: Strukturen
Heidelberg/Berlin/New York: Springer Verlag
1980 - 1983

/5/ Giloi, W.: Rechnerarchitektur
Heidelberg/Berlin/New York: Springer Verlag
1981

# ARBEITSWEISE EINER DATENFLUßARCHITEKTUR AM BEISPIEL DES µPD 7281

**Rieken, R., Dr.-Ing., KDT, Karl-Marx-Stadt, DDR**

Ein bedeutender Weg zur Steigerung der Verarbeitungsleistung von Rechnern ist die Orientierung auf Parallelsysteme. Da das Datenflußprinzip das einfachste und effizienteste der bekannten Verfahren zur Erkennung und Nutzung der impliziten Parallelität von Algorithmen ist /1/ und eine seiteneffektfreie Programmierung garantiert, ist es besonders gut geeignet zur Steuerung von parallel arbeitenden Verarbeitungsstrukturen.

Der seit etwa Anfang 1986 verfügbare Signalprozessor µPD 7281 der Fa. NEC /2/, /3/ ist nach Kenntnis des Autors der erste kommerziell verfügbare Datenflußprozessor als VLSI-Schaltkreis. Für diesen Prozessor wurde eine Entwicklungsumgebung zur Untersuchung der Leistungsfähigkeit geschaffen /4/ bis /7/.

In Datenflußrechnern wird die Steuerung des Ablaufs der Algorithmen implizit aus dem mit der Berechnung verbundenen Datenfluß abgeleitet. Datenflußsprachen drücken die Datenabhängigkeiten eines Algorithmus direkt durch die Programmstruktur aus. Die grafische Darstellung eines Datenflußprogramms führt zu einem Datenflußgraphen.

Beim µPD 7281 erfolgt die Programmierung auf Assemblersprachniveau. Für den zu implementierenden Algorithmus wird zuerst ein Datenflußgraph entwickelt, der dann direkt in die Funktionsassemblersprache des Prozessors umsetzbar ist.

Die gerichteten Kanten des Datenflußgraphen repräsentieren Datenpfade (engl.: links), auf denen die Knoten (funktionalen Operatoren) unidirektional Daten austauschen (Bild 1). Alle Ausgänge eines Knotens sind mit den Eingängen anderer Knoten (Operatoren) verbunden. Ausnahmen bilden Knoten zur Kommunikation mit der Umwelt. Die funktionalen Operatoren verarbeiten die ankommenden Datenwerte (Funktionsausführung) und senden einen (oder mehrere) Datenwert(e) als Ergebnis aus. Die zwischen den Knoten ausgetauschten Datenwerte werden als Token bezeichnet. Ein Token trägt Dateninformation

(Wert) und Steuerinformationen. Im allgemeinen existieren nur Werte und keine Variablen im Sinne der bekannten imperativen Sprachen.

Die Steuerung des Ablaufs der Verarbeitung basiert auf einer Regel zur Aktivierung der funktionalen Operatoren (Zündregel). Ein Operator "zündet" (wird zur Funktionsausführung aktiviert), wenn an jedem seiner Eingänge ein Token existiert.

Der µPD 7281 unterstützt fünf Typen von Knoten:

(a) Verarbeitungsknoten

Diese Knoten führen elementare Verarbeitungsfunktionen aus (Addition, Subtraktion, Multiplikation, logische Operationen, Konvertierungen u.a.m.). Die Operatoren haben bis zu zwei Eingänge und im allgemeinen einen Ausgang. Verarbeitungsknoten mit zwei Eingängen enthalten stets eine QUEUE-Funktion, die die Pufferung der ankommenden Token auf den Links steuert (Bild 1).

(b) Verbindungsknoten

Ein Verbindungsknoten (-operator) hat einen Eingang und bis zu 16 Ausgänge. Diese Knoten sind erforderlich, da im allgemeinen keine Variablen existieren. Die m-fache Nutzung eines Wertes (Tokens) erfordert somit die Generierung von m Kopien (Bild 2).

(c) Steuerknoten

Steuerknoten bilden das Pendant zu den Programmsteueroperationen in imperativen Sprachen (GOTO, IF .. THEN ... ELSE, CASE). Im Gegensatz zur dort realisierten Modifikation des Programmzählers erfolgt beim Datenflußprozessor eine Steuerung des Datenflusses.

Fast alle Steueroperationen basieren auf dem Zählen der ankommenden Token. Die Token aus dem Eingangstokenstrom werden in Abhängigkeit vom aktuellen Operatorzustand und vom Typ des jeweiligen Tokens auf die Ausgabelinks verteilt.

(d) Knoten zur Verwaltung globaler Daten

Im internen Datenspeicher können eindimensionale Datenfelder definiert werden. Der effektive Zugriff auf diese Vektoren wird durch eine Reihe von Funktionen für zyklische Lese- und Schreiboperationen unterstützt.

(e) Ausgabeknoten

Die Kommunikation des µPD 7281 mit der Umwelt (weitere µPD 7281, externer Speicher, Host-CPU) erfolgt ebenfalls auf der Basis von Tokentransfers. Zu diesem Zweck existieren Operatoren, deren Ausgabetoken über den Ausgabedatenbus an die Umwelt ausgegeben werden.

Zur Abbildung des Datenflußgraphen existieren im Prozessor zwei Tabellen. In der 64 * 40 bit großen Funktionstabelle (FT) werden die Operationskodes gespeichert. Da in vielen Fällen zwei Operatoren in einer FT-Zeile Platz finden, können somit im allgemeinen mehr als 64 Operatoren abgebildet werden.

Die Link-Tabelle (LT) umfaßt 128 Worte zu je 16 bit. Die hier gespeicherten Daten beschreiben die Kanten des Datenflußgraphen, d.h. die Verbindungen (Links) zwischen den einzelnen Knoten. Insgesamt ist somit auf einen Prozessor ein Datenflußgraph mit 128 Kanten und mehr als 64 Knoten abbildbar.

Literatur:


/1/ Giloi, W. K.: Datenflußprinzipien in Rechnerarchitekturen / In: NTG-Fachberichte Band 80, VDE-Verlag, Berlin (W) 1982, S. 19 - 41

/2/ Zoicas, A.; Look, G.: Datengesteuerter Prozessor beschleunigt die Bildverarbeitung. / In: Elektronik, München 1985, H. 3, S. 39 - 43

/3/ -: µPD 7281 Image Pipelined Processor. Product Description, NEC Electronics (Europe) ,1986

/4/ Rieken, R.: Einige Aspekte der Datenflußarchitektur / In: Nachrichtentech., Elektron., Berlin 38 (1988) 5, S. 173 - 177

/5/ Rieken, R.: Entwicklungsumgebung für den Datenflußprozessor µPD 7281 / In: radio fernsehen elektronik, Berlin (im Druck)

/6/ Rieken, R.: Matrixmultiplikation mit einem Datenflußprozessor / In: Bild und Ton, Leipzig (im Druck)

/7/ Rieken, R.: Schnelle Bildverarbeitungseinheit für PC. / In: Bild und Ton, Leipzig (im Druck)

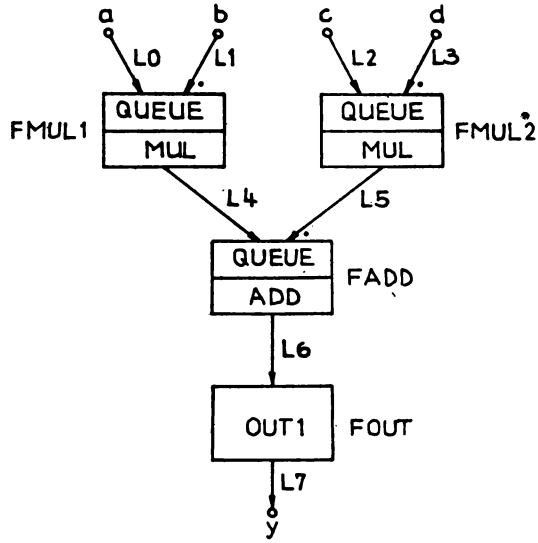



Bild 1 Datenflußgraph der Funktion $y = a \cdot b + c \cdot d$

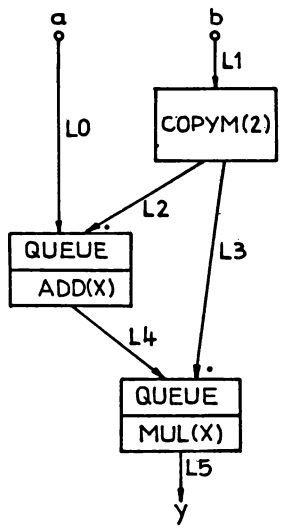



Bild 2: Datenflußgraph der Funktion $y = (a + b) \cdot b$

# ZUR ENTWURFSMETHODIK VON ENTWICKLUNGS- UND PROZESSRECHNER-SYSTEMEN.

Bräuning,P., Dr.-Ing., KDT, Ilmenau, DDR

Bei der Analyse des Begriffes "Entwurf" sowie all der Prozesse, die sich hinter dieser Bezeichnung verbergen, erscheinen gleichsam sowohl Eigenschaften und Begriffe wie Wissen, Algorithmus, formale Logik als auch solche, die sich mit Erfahrung oder Kreativität beschreiben lassen. Die Vielfalt und die scheinbare Widersprüchlichkeit des Entwurfsinhaltes wirft die Frage nach der Beherrschbarkeit eines solchen Komplexes an Begriffen und Fähigkeiten auf. Die Durchsetzung der Strategiemethodik des Entwurfs erfordert somit Prozesse, die über eine reine Datenverarbeitung hinausgehen und sich mit Wissens- oder gar auch Erfahrungsverarbeitung bezeichnen lassen. Aus der Menge und der Kompliziertheit dieser Prozesse ergibt sich die Forderung nach ihrer Ordnung und Unterstützung, um die Effektivität des Entwurfs zu erhöhen.

Effektive Verfahren des systematischen Rechnerentwurfs erlangen gegenwärtig eine stark zunehmende Bedeutung. Die dazu notwendigen Werkzeuge müssen Bestandteile von Arbeitsstationen sein, mit denen ein geschlossener Entwurf von Hardware und Software von der gedanklichen Entwurfsphase über den Test bis zur Inbetriebnahme unterstützt werden kann. Diese Systeme müssen bereits gegenwärtig - und perspektivisch immer stärker - Architekturentwurfssysteme sein und werden das qualitative Niveau von Expertensystemen sicher erreichen.

Modelle und Abstraktionsprinzipien sind beim Entwurf von mikroelektronischen Geräten und Systemen dann besonders praxiswirksam, wenn sie sich an der entsprechenden Entwurfssystematik orientieren:
Es erfolgt eine Zuordnung zwischen den Arbeitsniveaus und den virtuellen Maschinenniveaus, die hierarchisch aufeinander aufbauen. Der Übergang von einem Niveau zum nächsthöheren ist durch neue quantitative und qualitative Merkmale bestimmt.
Mit der Realisierung von Architekturfachsprachen in der Zuordnung zu den Architekturniveaus entsteht das Problem, daß eine

Vielzahl von Sprachebenen zur
- Verifikation,
- Spezifikation,
- Simulation und
- Emulation

in aufeinander abgestimmter Weise ausgewählt bzw. neu geschaffen werden müssen.
Architekturfachsprachen müssen den Regeln der formalen Sprachen genügen. Folgende Schlußfolgerungen sind damit ableitbar:
- Jede Ebene $m_1$ zur Beschreibung des Systemverhaltens ist primär durch die Auswahl einer Menge von:
  + Strukturelementen,
  + Verknüpfungsrelationen und
  + Informationstypen

  gekennzeichnet.
- Jede Ebene $m_1$ ist damit sekundär durch:
  + eine Menge von Verknüpfungsoperationen,
  + einen Homomorphismus der Verknüpfungsrelationen
  + und damit durch einen entsprechenden Homomorphismus der Verknüpfungsoperationen der Ebene $m_1$ bezüglich der Ebene $m_j$

  gekennzeichnet.
- Für jede Ebene entstehen Modelle.
- Initialmodelle beziehen sich immer auf Spezifikationen, die zu den Spezifikationen eines anderen Modells disjunkt sind.
- Ein Homomorphismus der auf die Relationen einer Spezifikationsebene angewendeten Operationen ist in jedem Falle dem Homomorphismus der Relationen der übergeordneten Spezifikationsebene unter Anwendung der entsprechenden Operationen identisch.

Unterschiedliche Beschreibungsmittel, Realisierungs- und Analyseniveaus besitzen dann besondere Effizienz, wenn die Ergebnisse eines Niveaus in das folgende Niveau überführbar sind. Das schließt eine Abhängigkeit von:
- Methode,
- Werkzeug,
- Sprache,
- Technologie und
- Organisationsform

voneinander ein.

Dem tragen folgende Entwurfsprinzipien Rechnung:
- Hierarchiebildung,
- Modularisierung,
- Aufgabenteilung,
- Modulkopplung und Strukturierung,
- Lokalität,
- Alternation zwischen Entwurf und Test,
- verwertbare Dokumentation und
- Vielfachverwendbarkeit.

Diese Prinzipien wurden an verschiedenen Komponenten untersucht und als relevant erkannt.

Universelle Entwicklungssysteme bilden die Voraussetzung für den systematischen Rechnerentwurf. Gerätetechnische Grundlage solcher Systeme sind leistungsfähige Mikrorechner, wie sie in Form von 16/32-Bit-Personal-Computern oder speziellen Entwicklungsrechnern verfügbar sind. Für die spezifische Funktion als Entwicklungssysteme werden diese durch eine Vielzahl von entwurfsunterstützender Software und durch Spezialbaugruppen zur Analyse und Emulation unterstützt und qualifiziert.
Diese mit "In-Circuit-Emulatoren" bezeichneten Einheiten bilden im Zielsystem den Prozessor nach, wobei komfortable Möglichkeiten der Steuerung und Auswertung der Tests realisiert werden:
- die emulierten (nachgebildeten) Prozessoren besitzen zeitlich, elektrisch und logisch Originalverhalten,
- alle Schnittstellen zum Zielsystem fungieren als CPU-Adapter,
- als Hauptfunktionsgruppen sind Emulatorspeicher, Seitensteuerung, Haltepunktlogik, Historyspeicher und Prozessor-Steuerlogik enthalten,
- die Kopplung zum Entwicklungsrechner wird über Standardschnittstellen realisiert,
- im Entwicklungsrecnner existiert ein leistungsfähiger Monitor,
- es werden mehrere Zielprozessoren unterstützt.

Gegenwärtig wird die ursprüngliche Aufgabe von Entwicklungssystemen als "Simulatoren und Emulatoren" von Zielsystemen grundsätzlich zu Arbeitsstationen erweitert. Das bedeutet eine Unterstützung des Rechnerentwurfes in folgenden Punkten:
- syntaktische und semantische Formulierung der Entwurfsaufgabe,
- Analyse der Aufgabe,

- logischer Entwurf,
- Schaltungsentwurf,
Leiterplatten-Layout,
- Testung der Hardware,
- Entwurf und Testung der Software,
- Testung der Zielsoftware auf der Zielhardware.

Auf Grund dieser Aufgaben entwickeln sich seit einigen Jahren Systeme heraus, die als Arbeitsstationen (Work Station) bezeichnet werden und die gegenwärtig immer stärker das methodische Niveau von Expertensystemen erreichen.
Dabei übernimmt die Expertensystem-Schale zunächst folgende Aufgaben:
- Wissensrepräsentation im Bereich des Rechnerentwurfs,
- Umgang mit und Bewertung dieses Wissens,
- Verwendung von Heuristiken und Fehleranalyseverfahren,
- Einbeziehung von Faktenänderungen während des Problemlösevorganges,
- Interpretation mehrdeutiger Daten.

Beim Entwurf eines Entwicklungs-Expertensystems muß davon ausgegangen werden, daß die meißten Komponenten des klassischen Rechnerentwurfs weiter genutzt werden. Auf diese bewährten Schalen wird das Expertensystem aufgesetzt. Die Algorithmen zum Entwurf und zur Inbetriebnahme sind damit nicht mehr direkt erkennbar, sondern werden durch einen Regelapparat überlagert.

Der erreichte internationale Stand läßt erkennen, daß sich der rechnergestützte Entwurf immer stärker durchsetzt und die Entwicklungssystemtechnik einen Stand erreicht hat, der weitgehend den heutigen Anforderungen in Bezug auf Komplexität gerecht wird. Eine direkte Kopplung von Entwicklungssystemen, Prozeßrechnern und Expertensystemen ist notwendig, da die Fülle an sehr unterschiedlichen Entwurfs- und Testinformationen sowie möglicher Entwurfswege zu groß ist, um determiniert oder heuristisch ohne eine Rechner-Wissensrepräsentation und -verarbeitung schnell zum optimalen Zielsystem zu gelangen. Expertensysteme als höchste Stufe von Beratungssystemen haben sich in vielen Bereichen des rechnergestützten Entwurfs und Tests bereits einen guten Stand errungen.

# ВЫСОКОПРОИЗВОДИТЕЛЬНЫЙ КОМПЛЕКС НА ОСНОВЕ ПРОЦЕССОРА ДЛЯ МАТРИЧНЫХ ВЫЧИСЛЕНИЙ И 16-РАЗРЯДНОЙ МИНИ ЭВМ ТИПА СМ-4

Симеонов Б., Каишева В., Гелиболян Н., Висулчева Д., Тодоров Л., София/НРБ

Как известно, 16-ти разрядная мини ЭВМ типа СМ-4 получила широкое распространение и для ее создано развитое прикладное программное обеспечение. Однако, она имеет ограниченную оперативную память (256 Кбайта) и невысокую производительность (порядка 200 тыс.операций/сек.).

Для матричных вычислений эти недостатки компенсируются подсоединением матричного процессора (МП) к мини ЭВМ. Таким способом создан высокопроизводительный комплекс (ВПК) ИЗОТ 1085 С. Он включает 16-разрядную мини ЭВМ ИЗОТ 1016С.М1 типа СМ-4 с интерфейсом "Общая шина", периферийный высокопроизводительный матричный процессор ИЗОТ 2001С и большую память на накопителях на магнитных дисках (НМД) 200 Мбайта (см. черт. 1).

Матричные процессоры предназначены для все более расширяющегося круга задач, связанных с цифровой обработкой больших массивов данных, накопленных во время экспериментов и наблюдений и в процессах проектирования и моделирования в современной науке и технике. Таким образом ВПК дает пользователю возможность создавать специфические для него обрабатывающие функции.

В этой цели для матричного процессора предусмотрены: базовое программное обеспечение (БПО для МП); программная система для разработки, отладки и редактирования прикладного программного обеспечения (СРПМП); тестовое программное обеспечение (см. черт. 2). Все эти продукты созданы на широко известном языке высокого уровня ФОРТРАН-77.

ВПК состоит из центральной части и терминальной подсистемы. В центральную часть включены центральный процессор ИЗОТ 2104С.М1 типа СМ-4П, матричный процессор ИЗОТ 2001С, ОЗУ ИЗОТ 3502С.М1. 5423-256 Кбайта, пульт оператора СМ 6326, мини-накопитель на магнитной ленте СМ 5309-2шт, развитая внешняя память, включающая 2 шт. НМД емкостью 200 Мбайта. Принципиально возможно включение 8 штук НМД.

ВПК имеет коммуникационные возможности на основе асинхронных адаптеров ИЗОТ 6003С.М1.5512 и мультиплексоров ИЗОТ 8506С,

синхронных адаптеров ИЗОТ 8504С.М1, статистического мультиплексора СМ 8520 и контроллера для локальной сети ИЗОТ 8505С.

ВПК может работать как универсальная ЭВМ типа СМ4 и, кроме того, благодаря наличию матричного процессора ИЗОТ 2001С, обеспечивает пользователям мини ЭВМ производительностью больших ЭВМ. Производительность МП-12 млн. операций/сек. в формате плавающей запятой.

Используемый матричный процессор ИЗОТ 2001С обладает следующими особенностями:

- имеет управляющую память, (с 64 разрядными словами) содержание которой не фиксировано твердо определенным количеством обрабатывающих функций, а может перезагружаться из библиотеки, содержащей свыше 400 заголовков;

- имеет собственную оперативную память (с 32 разрядными словами) для данных объемом порядка 1-го Мбайта, которая делает его вполне независимым от основной ЭВМ во время вычислений всего пользовательского алгоритма обработки; в этой же цели в МП встроены также и постоянная и перезагружаемая память для таблиц и коэффициентов;

- обеспечивает пользователю возможность создавать специфические для него обрабатывающие функции; в этой цели предусмотрены: язык ассемблерского типа, транслятор, симулятор и диалоговый отладчик; для пользователей, которые не имеют возможности программировать в больших деталях, предусмотрено специальное подмножество языка высокого уровня ФОРТРАН - 77;

- имеет один конвейерный сумматор и один конвейерный умножитель;

- обеспечивает быстроту вычислений при помощи 2-ух быстродействующих буферов, работающих со скоростью конвейеров и обеспечивающих возможность 4-ех независимых обращений к данным в одном цикле машины;

- имеет самостоятельные узлы для целочисленной адресной арифметики, управления переходами и разветвлением в программе;

- имеет многомагистральную структуру внутреннего обмена, в которой одновременно возможны почти все внутренние междумодульные передачи, что существенно облегчает закладывание параллелизма на программы и освобождает аппаратуру для выполнения только максимально быстрой обработки.

Для центрального процессора (ЦП), МП является лишь одним из

устройств, подключенных к "Общей шине".

Например, если ЩШ (арбитратор шины) управляет "Общей шиной", он может извлекать данные из МШ. Если МШ запросит и получит управление шиной из ЩШ, то он будет в состоянии инициировать коммуникации с памятью или с другими устройствами интерфейса "Общая шина". Обмен больших блоков данных между МШ и памятью может занимать "Общую шину" непрерывно, при этом ограничивая доступ других устройств к шине. Поэтому в МШ предусмотрен режим последовательных циклов обмена. В этом режиме, МШ занимает последовательные циклы доступа к памяти и потом освобождает "Общую шину", чтобы и другие устройства имели возможность обмена. Количество этих последовательных циклов (1-16) выбирается аппаратно, при помощи перемычек, на платах управления интерфейсом.

Комплекс работает под управлением дисковой операционной системы реального времени ДОС РВ-Б, версия 04. Для выполнения пакетов прикладных программ нужно пользоваться языком высокого уровня "Языковый процессор ФОРТРАН 77".

Для комплекса разработан набор математических библиотек:

- библиотеки микропрограмм для обработки сигналов для МШ ИЗОТ 2001С, которая представляет собой процедурные пакеты, предназначенные для обработки цифровых сигналов, и реализирующие следующие операции: вычисление свертки, вычисление дискретного преобразования Гильберта и окна Хэнна, цифровую фильтрацию, анализ случайных процессов;
- библиотеки микропрограмм для геофизических исследований для МШ;
- библиотеки микропрограмм для обработки изображений для МШ;
- дополнения к библиотеке микропрограмм основных функции матричных процессоров для научно-технических задач, связанных средствами линейной алгебры, и для симулирования и моделирования в области механики, физики и т.д.

Микропрограммы составлены на языке АССЕМБЛЕР (АРАІ) для МШ 2001С и вызываются оператором САІІ на языке ФОРТРАН.

Пользователь имеет возможность на базе этого набора создать свои прикладные пакеты.

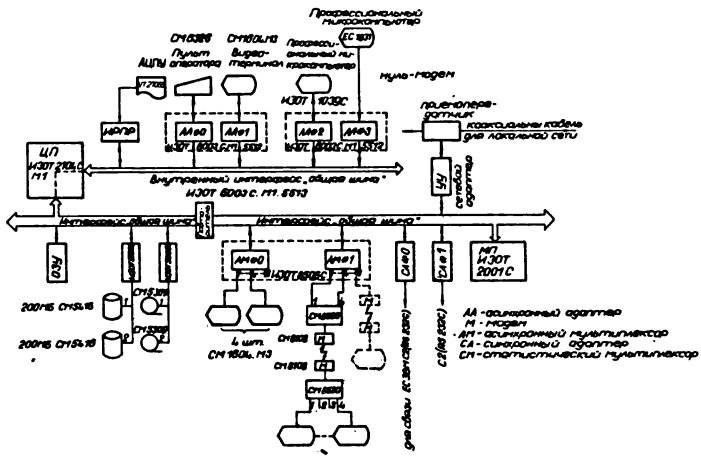
ВШК обеспечивает возможность эффективного решения задач обработки цифровых сигналов в следующих областях народного хозяйства: геофизике, сейсмических исследованиях, гидро - и радиолокации, звуковой и ультразвуковой дефектоскопии, цифровой обра-

ботке изображений, обработке речевых сигналов, исследования материалов, в технике связи и т.д.

## Л И Т Е Р А Т У Р А

1. Фрэнк Т., С.,PDP-11 Архитектура и программирование, М., "Радио и связь", 1986

2. Матричный процессор ИЗОТ 2001С, Техническое описание, С., ТХО "ИЗОТ", 1984




Черт. 1

Обозначения:

| | |
|---|---|
| ДОС РВ-Б | Операционная система для МЕИМ СМ4 |
| БПО для МП | Базовое программное обеспечение для МП ИЗОТ 2001С. |
| EPSLIB | Программный интерфейс между МЕИМ и МП ИЗОТ 2001С. |
| HSRLIB | Основная математическая библиотека |
| СРПМП | Система для разработки, редактирования и отладки программ для МП ИЗОТ2001С |

Черт. 2

# EINBAU EINER ECHTZEITUHR UND EINES OPERATIONSSPEICHERS MIT STROMUNABHÄNGIGER VERSORGUNG BEI 16-bit PERSONALCOMPUTERN UND EINE METHODE FÜR IHRE STEUERUNG

Vitanov, B., Ing., Sofia- CIITT/VRB
Petrova, N., Dr., Ing., Sofia - CIITT/VRB
Karalamov, E., Ing., Sofia - CIITT/VRB

## VORAUSBESTIMMUNG

Der Modul enthält CMOS RAM,im Personalcomputer als Operationsspeicher benutzt.Die Echtzeituhr vereinigt in sich drei unabhängige Anwendungen:eine komplette Uhr mit Alarmruf für eine bestimmte Stunde,ein Kalender für 100 Jahre und Fertigung programmierter periodischer Unterbrechungen.

Bei Ausfall der Hauptversorgungsspannung gewährleistet der Modul die Erhaltung der Information im Personalcomputer.Umschaltende Logik garantiert die Versorgung des Operationsspeichers und der Echtzeituhr von einer Batterie und bereitet beim Ausfall der Hauptversorgungsspannung ein Verbotssignal vor.

## STRUKTURELLES SCHEMA DES MODULS, Bild 1

Der Modul "CMOS RAM mit Echtzeituhr" besteht aus folgenden Baueinheiten:

Echtzeituhr mit Kalender

Die Echtzeituhr mit Kalender ist durch integrierte Schaltung MC 146818 [1] realisiert worden.Sie enthält 50 Bytes RAM, an den der Mikroprozessor während des Aktualisierungszyklus UIP (update in progress) völlig Zugriff hat,10 Bytes - mit normaler Aufbewahrung von Zeit,Kalender und Alarm und vier Kontrollregistern und Zustandsregistern (A, B, C, D).

Bit 7 aus Register A enthält Statuflag UIP,das vom Programm benutzt wird.Zugriff an die Information in RAM der Echtzeit uhr,des Kalenders und des Alarms hat der Prozessor nur bei UIP=0.An die 10 Informationsbytes (für Zeit,Kalender und Alarm) hat das Programm nicht immer Zugriff.Sie wreden von der Aktualisierungslogik umgeschaltet,um mit 1 Sek.erhöht und nach einer Alarmbedingung geprüft zu werden. Während dieser Zeit ist bit UIP=1. Dieser bit kann auf Null eingestellt werden, wenn bit SET=1 im Register B gesetzt wird.

Umschaltende Logik

Damit die Information in CMOS RAM und in RAM der Echtzeituhr erhalten bleibt,ist es notwendig,dass bei Ausfall der Hauptversorgungsspannung für eine Speisespannung mit zulässigem Wert zur Aufbewahrung des Betriebes STAND BY gesorgt wird.Zu diesem Zweck werden die Versorgungseingangsleitung von CMOS RAM,die integrierten Schaltungen und die Versorgungseingangsleitung der Echtzeituhr in eine Pufferschaltung,die beide Arten von Werten der Spannung ermöglicht,eingeschaltet,in Abhängigkeit von dem Betrieb - beim Betrieb STAND BY ist dieser Wert niedriger.Die einfachste Realisierung einer solchen Schaltung ist die Verbindung aller Versorgungseingänge in "Dioden ODER".In diesem Fall wird eine normale Speisung für Betrieb STAND-BY ermöglicht,aber die Versorgungsspannung bei dynamischem Betrieb ist niedriger infolge der Senkung der Diodenspannung bei "Dioden ODER".Auf dem Bild 2 ist eine Schaltungslösung gezeigt,bei der die Diode in "Dioden ODER" mit einem Schalter, der auf die Senkung der Hauptspannung reagiert und der minimale Spannungssenkung gewährleistet, ersetzt wird.Der Schaltertransistor T1 arbeitet im Betrieb der vollen Sättigung und gewährleistet minimale $U_{ce\,sat}$. Als Schaltertransistor ist die Anwendung von VMOS Transistoren möglich,die beim durchlässigem Kanal eine Senkung unter 0,1V gewährleisten.

Nach dem Ausfall der Hauptversorgungsspannung und nach der Einführung vom Betrieb STAND BY sollen die Aufzeichnung und das Lesen von CMOS RAM und die Echtzeituhr eingestellt werden.

Auf dem Bild 3 ist eine Blockschaltung zur Erzeugung von Verbotssignalen für den Speicherzugriff beim Ausfall der Versorgungsspannung [2] gezeigt.

Damit der Mikroprozessor die angefangene Operation vor dem Ausfall der Hauptversorgungsspannung fortsetzt,ist noch etwas Zeit für die Beendigungsoperationen zur Aufbewahrung von Zwischenergebnissen notwendig.Zu diesem Zweck wird bei Senkung der Versorgungsspannung auf den kritischen Wert $U_K$,Bild 4,durch ein Komparatorsignal unmaskierte Unterbrechung NMI an den Mikroprozessor gesendet,damit die Beendigungsoperationen anfangen.Bei der Ausführung dieser Operationen sendet der Mikroprozessor Signale für Auswahl,Aufzeichnung und Lesen an CMOS RAM und an die Echtzeituhr mit RAM.Nach einer Zeit $t_{BO}$ , die zur Ausführung von Beendigungsoperationen notwendig ist,gibt der Mikroprozessor ein Signal,das

in Übereinstimmung mit NMI vom Komparator ein Signal RESET er - zeugt,die Prozessorarbeit verbietend und die Wendung an CMOS RAM und an die Echtzeituhr blockierend.

CMOS Operationsspeicher

Der stromunabhängige Operationsspeicher ist mit integrier ten Schaltungen TC 5564 durch Organisation 8K x 8 [3] realisiert. worden.In einem Modul können 16 integrierte Schaltungen eingebaut werden,indem der Volumen des Operationsspeichers 128 KBytes ist. Es besteht eine Möglichkeit zur Einstellung der Anfangsadresse des Operationsspeichers durch 128 KBytes bei Anwendung mehrerer Module CMOS RAM in der Konfiguration des Personalcomputers.

BEISPIEL AM ALGORITHMUS EINES SERVERPROGRAMMS FÜR DIE UHR

Am Bild 5a ist ein Algorithmus eines initialisierenden Programms für die Uhr,das periodisch ein unterbrechendes Signal mit einer Periode 480 $\mu$s bei Basisfrequenz 32 kHz,in einem binär-dezimalen Datenformat in den 10 Registern für Zeit,Kalender und Alarm und mit 24-Stundenformat für die Stundenregister erzeugt.

Auf Bild 5b ist ein Algorithmus eines Teils des Programms für die Verarbeitung von unterbrechenden Signalen,die die Unterbrechung der Uhr kontrollieren,gezeigt.

KONSTRUKTIVE GESTALTUNG DES MODULS

Konstruktiv ist der Modul als eine 8-schichtige Leiterplatte mit Ausmassen 220mm x 144,45mm gestaltet.Zur Schutz des Moduls vor äussere und inneren Störungen ist "ERDEN" der Leiterplatte als eine gesonderte Schicht realisiert worden.Der Modul wird mit der systeminternen Schnittstelle des Mikrocomputers [4] durch indirekte Kupplung Type RFT 113 verbunden.

Literatur

1. Motorola semiconductors microprocessors data manual,1982
2. Karalamov E. u.a. "Elektronische Kasseneinrichtung",Autorenzeugnis Nr.40787,4.12.1985
3. Mos memory products data book 1982 Toshiba corporation
4. Petrova N., Spezialisierte digitale Rechenmacshine,wissenschaftliche Tagung - 9-10 Juni, Schumen, 1988

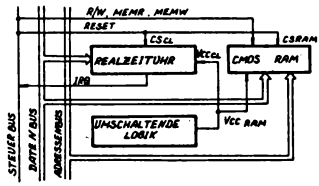



BILD 1

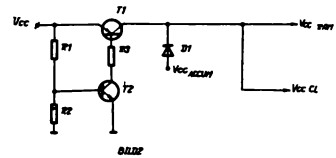



BILD2

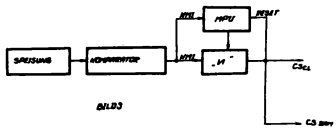



BILD3

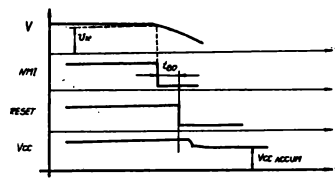



BILD4

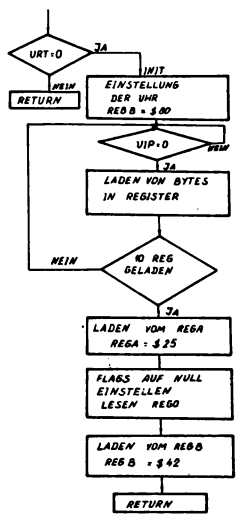



BILD 5a

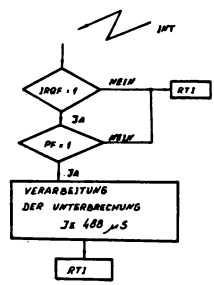



BILD 5B

ENTWURFSSYSTEM FÜR VLSI-SPEICHER

Knobloch, J., Dr.-Ing., KDT, Dresden, DDR

1. Allgemeines

Mit dem Übergang zu minimalen Strukturbreiten von 1 µm und darunter sind Chips mit über 1 Million Bauelementen realisierbar. Der Vergleich der Zielstellungen bei der Entwicklung von Speicherschaltkreisen und Logikschaltkreisen dieser Größenordnung zeigt, daß unterschiedliche Entwurfstechnologien erforderlich werden. Mega-Speicher:

- Schaltkreise auf Basis von Millionen gleichartiger Transistoren und Kapazitäten in regulärer Anordnung und Verdrahtung.

- Jede Speicherzelle hat die gleiche Funktion.

- Speicherschaltkreise werden in großen Stückzahlen hergestellt. Deshalb ist eine umfassende Entwurfsoptimierung zur Sicherung der Ausbeute notwendig.

Mega-Logik:

- Schaltkreise mit bis zu einigen 100 000 verschiedenen Transistoren unterschiedlicher Anordnung und Verdrahtung entsprechend der Vielfalt und Komplexität der anwendungsspezifischen Lösung.

- Die Funktionsvielfalt wird mit einer großen Anzahl verschiedener Zellen -von einfachen Logikgattern bis zu komplexen Macrozellen- realisiert. Dabei ergeben sich unterschiedliche Chipgrößen mit individueller Pinzahl.

- Trotz des wachsenden Bedarfs an ASICs insgesamt ergeben sich auf Grund der Typenvielfalt vergleichsweise kleine Produktionsstückzahlen.

Die sichere Entwicklung dieser Chips ist nur mit umfangreicher Rechnerunterstützung in Form von leistungsfähiger Hardware und vielfältigen Programmen möglich. Die Einheit von Entwurfstechnologie und Rechnerunterstützung wird im weiteren als Entwurfssystem bezeichnet.

Die gemeinsamen Anforderungen an Entwurfssysteme zur Entwicklung von Mega-Speichern und Mega-Logik sind:

- Verarbeitung der großen, mit dem Integrationsgrad verknüpften Datenmengen in allen Entwurfsebenen,

- Sicherung der Entwurfsqualität in Einheit von Ausbeutesicherheit und Fehlerfreiheit der Entwürfe,

- Erhöhung der Entwurfsproduktivität und damit Senkung der Entwicklungszeit.

2. Besonderheiten des Entwurfs von VLSI-Speichern

VLSI-Speicher sind eine Schaltkreisklasse mit besonderen Eigenschaften hinsichtlich Integrationsgrad, Regularität und universellem Einsatzgebiet. Außerdem fungieren sie meist als Prototyp oder Leitprodukte für die Entwicklung neuer Technologien, so daß -im Gegensatz zu anderen Schaltkreisentwicklungen- die Entwurfsgrundlagen erst im Prozeß der Anpassung Entwurf-Technologie während der Entwicklung entstehen. Aus beiden Gesichtspunkten heraus ergeben sich erhebliche Besonderheiten der Entwurfstechnologie für VLSI-Speicher.

VLSI-Speicher sind aus drei Gründen heraus als Prototypen für neue Technologien prädestiniert:

- aufgrund ihres hohen Integrationsgrades und weiterer extremer Parameter ergeben sich höchste Anforderungen an die Herstellungsverfahren bzgl. Strukturbreiten, Schichtdicken, Ebenenzahl und Defektdichten;
- durch die regelmäßige Struktur erleichtern sie den Schaltungsentwurf und die Fehleranalyse, was zur Senkung des Entwicklungsrisikos und zur beschleunigten Schaltungsoptimierung beiträgt;
- durch Massenfertigung der Speicherschaltkreise wird ein hohes und stabiles Ausbeuteniveau erreicht - Basis für die sichere Produktion weiterer Schaltkreise in der entsprechenden Technologie; dabei wird die erreichte Leistungsfähigkeit der Tech-

nologie umgesetzt in Spitzenparameter (Integrationsgrad, Geschwindigkeit, Leistungsverbrauch) oder indirekt in eine Ausbeuteverbesserung (Chipfläche, größere Toleranzfenster).

Für den Entwurf von Speicherschaltkreisen ist eine enge Verbindung zur Technologieentwicklung kennzeichnend. Als Schwerpunkte sind seitens des Entwurfs zu bearbeiten:

- der eigentliche Schaltungsentwurf,
- Entwicklung und Einsatz eines ausgereiften CAD-Entwurfssystems
- und die Schaffung der physikalischen Entwurfsgrundlagen.

3. Die Entwurfstechnologie für VLSI-Speicher

Die Entwurfstechnologie muß es gestatten, Speicherschaltkreise in unterschiedlichen Technologien und in den perspektivischen Integrationsgraden vor allem als Prototypentwicklung zu bearbeiten. Dabei steht das Ziel, die Entwicklung eines Speicherschaltkreises in drei Entwurfsiterationen abzuschließen:

- Der 1. Entwurf bringt funktionsfähige Muster.
- Der 2. Entwurf liefert pflichtenheftgerechte Bauelemente.
- Der 3. Entwurf ist ein optimierter Entwurf zur Sicherung einer hohen Produktionsausbeute.

Bei der Prototypentwicklung hängt das Erreichen dieser Zielstellung vom Vorlauf der Technologieentwicklung ab. Bei der zumeist praktizierten Parallelentwicklung von Technologie und Speicherschaltkreis können die technologischen Parameter bis zum Entwicklungsabschluß korrigiert bzw. aktualisiert werden, zum Beispiel beim Einsatz neuer Ausrüstungen. Damit ändern sich die Entwurfsgrundlagen und die nachgewiesene Funktion des Speichers ist erneut zu bestätigen. Weitere Aufgaben ergeben sich für den Schaltungsentwurf nach Ablauf der Entwicklung, wenn im Rahmen der Typpflege und nach Auswertung der ersten Großserien Maßnahmen zur Ausbeutesteigerung eingeleitet werden, die auch eine Entwurfsüberarbeitung beinhalten können. Der Entwurf von VLSI-Speichern (Prototypen) erfolgt in den vier Etappen

- Schaffung der Voraussetzungen
- Entwurfsvorbereitung

- Schaltungs- und Layoutentwurf
- Verifikation und Kontrolle

4. Rechnerunterstützung des Entwurfs von VLSI-Speichern

Voraussetzung für den Entwurf höchstintegrierter Speicher ist eine effektive, zugeschnittene Software. Hauptziel ist die

- Beherrschung der großen Datenmengen bei Speichern und bei den für die Technologieentwicklung unentbehrlichen Testfeldern,
- Verkürzung der Rechenzeiten und
- ein trivialfehlerfreier Erstentwurf.

Generell wird dieses Ziel angestrebt durch eine
- durchgängige einheitliche Hierarchie in allen Entwurfsebenen,
- weitgehend grafische Arbeitsweise mit dem Farbrasterdisplay,
- umfassende Rechnerunterstützung bei den Routineprozessen der Datenträgerherstellung und
- Zusammenführen der Einzelprozesse durch Datenverwaltung.

Schwerpunktmäßig kommen folgende Programme zum Einsatz:

- Prozeß- und Bauelementesimulation
- Auswerte- und Optimierungsprogramme zu den Bauelementemodellen
- Netzwerkanalyse
- Eingabe und Simulation von Schaltungen und Layout
- Layoutverifikation
- Layoutdatenverarbeitung und Ausgabe der Ansteuerdaten für die Bildgeneratoren.

Für
- Automatische Generierung des Rahmen- und Markensystems
- Erweiterung der Verifikation um Prüfung der elektrischen Entwurfsregeln
- Spezielle Software für Testfeldentwurf und Auswertung
- Schablonenverifikation und
- Chipverifikation

sind weitere Programme im Rahmen des Entwurfssystems für VLSI-Speicher zu entwickeln.

# DAS MODULARE MIKRORECHNERSYSTEM S 4000

Fengler, W., Dr.-Ing., TH Ilmenau, DDR
Kowalski, O., Dr.-Ing.
Roth, M., Prof.Dr.sc.phil.Dr.-Ing.

## 1. Systemeinordnung

Modulare Systeme bilden einen Einsatzschwerpunkt der Mikrorechentechnik. Dabei sind Mikrorechnermodule funktionell weitgehend in sich geschlossene, ueber standardisierte Schnittstellen koppelbare Baugruppen. Als Hauptvorteile ihres Einsatzes koennen genannt werden:

- Flexibilitaet beim Einsatz
- Reduzierung der Einzweckloesungen durch standardisierte Baugruppen und effiziente Entwicklung und Fertigung
- Verringerung des Aufwandes zur Systementwicklung und -inbetriebnahme durch Teilloesungen mit gesicherter Funktion.

Der Einsatz eines modularen Automatisierungssystems ist technisch und oekonomisch von Vorteil, wo funktionell und konstruktiv hohe Flexibilitaet durch Auf- und Abruestung bei kleinen Fertigungslossgroessen erforderlich ist. Dadurch wird bei hoher Systemzuverlaessigkeit und Verfuegbarkeit eine hohe Anpassung an den jeweilig zu automatisierenden Prozess erreicht.

Das modulare Mikrorechnersystem S 4000 /1,2/ ist auf Basis der Einchip-Mikrorechner-Schaltkreisfamilie (EMR) U 881 und folgende konzipiert, wobei der Einsatzschwerpunkt in der prozessnahen Ebene der Klein- und Mittelautomatisierung liegt. Durch den modularen Charakter und die standardisierten Schnittstellen im System und nach aussen ist es moeglich, eine voellig funktionelle Dezentralisierung zu erreichen. Dies gilt auch fuer oertliche Dezentralisierung unter bestimmten Randbedingungen wie Entfernung, Einsatzbedingungen,

konstruktive Bedingungen, Service und Inbetriebnahme. Dabei kann die Anwendung

- einzelne Steuerungs- oder Regelungsaufgaben
- Maschinen- oder Aggregate-Steuerung bzw. Regelung
- Geraeteintegrierter Rechner zur Messwertverarbeitung
- Loesung von Automatisierungsaufgaben in der untersten Ebene in Zusammenhang mit einem hierarchischen System

umfassen.

Daraus resultieren die wesentlichen Anforderungen fuer die Gestaltung der Systemarchitektur:

- Hoehere Verarbeitungsleistung als Universal-CPU-Loesungen bei gleichem bzw. geringerem oekonomischen Aufwand.
- Einsatz in Einzelaufgaben bis zur Integration in Geraete und Anlagen. Die Module sind weitgehend autonom.
- Anwendung von Teilloesungen in Form von OEM-Baugruppen. Die Module loesen typische Aufgaben der dezentralen Prozessdatenverarbeitung, sowohl als reine Hardware-Loesung als auch Software unterstuetzte Loesung.
- Anwender-Software-Entwicklung ueber dem Maschinenniveau. Der Software-Entwurf wird je Modul mit einer oder mehreren problemnahen Programmiersprachen unterstuetzt.
- Hohe Systemzuverlaessigkeit durch gezielte Redundanz, weitgehende Autonomie und entsprechende Schnittstellen.

## 2. Interner serieller Systembus ISB 881

S 4000 ist ein modulares Mehrrechnersystem mit weitgehend fester Aufgabenteilung. Das globale Betriebsmittel zur Modul-Kommunikation ist der Interne serielle Systembus ISB 881. Der ISB 881 gewaehrleistet die aus der Aufgabenklasse Prozessdatenverarbeitung folgenden Eigenschaften:

- Multimaster-Faehigkeit
- Definierte Buszugriffszeiten
- Rekonfigurierbarkeit bei Teilnehmerausfall bzw. -einfuegung
- Datensicherung.

Durch die Auslegung des ISB 881 als Komponentenbus /3/ wurden zwei Protokollebenen realisiert, die physikalische und die Kommunikationsebene.

Die physikalische Ebene basiert auf der 20 mA-Stromschleife und einer Linienstruktur aller Teilnehmer mit folgenden Eigenschaften:

- Einfache galvanische Trennung fuer Sender und Empfaenger
- Mehr als ein aktiver Teilnehmer durch verdrahtetes ODER
- Eigenfunktionsueberwachung durch Mithoeren der Sendung
- Hardwareseitige Kopatibilitaet des ISB zum IFSS-Interface.

Die Kommunikationsebene ermoeglicht durch Adress-, Typangaben und Sicherungsfelder im Uebertragungsprotokoll den Austausch von Informationen zwischen zwei Busteilnehmern sowie Fehlererkennung und -behandlung.

Beim fuer den ISB 881 angewendeten Token-Passing-Verfahren sind die Masterfunktion, die Kontrollfunktion und die Leitfunktion dezentralisiert. Damit wird Multi-Master-Faehigkeit und Rekonfigurierbarkeit im System gewaehrleistet.

## 3. Modul- und Systemarchitektur

Die Systemarchitektur des S 4000 basiert auf zwei Modul-Grundtypen:

- Peripherie-Module und
- Verarbeitungsmodule.

Peripherie-Module erfuellen die Aufgaben:

- Direkte Kopplung der Mess- und Stellglieder des Prozesses ueber digitale und analoge Schnittstellen
- Kommunikation Bediener-Prozess in unmittelbarer Prozess- bzw. Geraeteebene
- Ankopplung von Mess-, Labor- und Peripheriegeraeten ueber Standard-Schnittstellen
- Kopplung zu Leitrechnern ueber Standard-Schnittstelle
- Primaere Prozess- und Kommunikationsdatenverarbeitung mit DDC von Geraeten und Teilanlagen.

Verarbeitungsmodule haben die Aufgaben:

- Systeminitialisierung
- Verknuepfung der Daten mehrerer Peripherie-Module in einer uebergeordneten Hierarchieebene
- Systemkopplung zu einem Leitrechner
- Erzeugung eines zweiten ISB bzw. Standardbussystems
- EPROM-Programmierung bei Systemgenerierung.

Die Moduln enthalten die auf die Aufgabenklasse zugeschnittene Firmware, die mit dem Anwenderprogramm (Fachspraohniveau) die konkrete Verarbeitungsaufgabe loest.

4. Software-Architektur

Die vorgesehenen Haupteinsatzfaelle der MMS 881-Module sind:

- OEM-Einsatz der Module mit modulbezogener Software
- Systemeinsatz unter Nutzung modulbezogener Software und
- Systemeinsatz durch Konfigurierung und Parametrierung vorverabschiedeter Module.

Auf dem S 4000-Moduln werden aufgabenorientierte Firmwareinterpreter zur Abarbeitung von Zwischenkodes von Fach- oder universelllen hoeheren Programmiersprachen implementiert. Es

wurden Sprachen zur analogen Signalverarbeitung und Regelung (PROMAR /4/), zur Steuerung (MPSS /5/), zur Dialogprogrammierung /6/ und zur allgemeinen Informationsverarbeitung (FORTH /7/) eingesetzt.

Fuer die Erstellung der Anwenderprogramme wird ein Entwicklungssystem (A 5120, P 8000 mit Betriebssystem UDOS) erstellt, das folgende Komponenten beinhaltet:

- Uebersetzer FIG-FORTH - FORTH 8
  Simulator FORTH 8
- Uebersetzer MPSS-Quelle - MPSS 8
- Uebersetzer PROMAR-Quelle - PROMAR 8
- Monitor fuer In-circuit - Emulator (/8/).

## 5. Module im System S4000

| Modulbezeichnung | Funktion |
|---|---|
| Verarbeitungsmodul | - Systeminitialisierung und -datenverarbeitung<br>- Erzeugung und Verwaltung von zwei Bussen ISB oder eines ISB und einer asynchronen seriellen Schnittstelle (V24, IFSS) |
| Reglermodul | - Einkanalige PI-Reglerfunktion<br>- Stellgliedsteuerung und -ueberwachung<br>- Minimalmonitor zur Bedienung und Anzeige<br>- DDC und Sollwertfuehrung vom ISB<br>- Einsatz im S4000-Verbund oder autonom |
| Analogmodul 1 | - Erfassung und Ausgabe von Analogsignalen<br>- Allgemeine Signalverarbeitung einschliesslich Regelung (PROMAR) |

| | |
|---|---|
| | - Signalkommunikation ueber ISB im S4000-Verbund |
| Steuerungsmodul 1 | - Erfassung und Ausgabe von Binaersignalen<br>- Steuerungstechnische Signalverarbeitung (MPSS)<br>- Signalkommunikation ueber ISB im S4000-Verbund |
| Parametrier- und Strukturiereinheit | - Anzeige-, Bedien-, Parametrier- und Strukturierung von Analog- und Steuerungsmodule<br>- Test-, Eingabe- und Inbetriebnahmeeinheit speziell fuer Steuerungsmodule bzw. -programme |

Tabelle 1: Aktive Module

| Modulbezeichnung | Funktion |
|---|---|
| E/A-Erweiterungsmodul 1 (zum Steuerungsmodul) | - E/A-Erweiterung auf 24 Eingaenge 8 Ausgaenge |
| E/A-Erweiterungsmodul 2 (zum Steuerungsmodul) | - E/A-Erweiterung auf 16 Eingaenge 16 Ausgaenge |
| Relaismodul 1 | - 8 Relais GBR 12 (Leistungsschalter) |
| Relaismodul 2 | - 12 Relais GBR 12 (Leistungsschalter) |
| Stromversorgungsmodul | - 24V Rohspannung aus 220V Netzspannung fuer 7...10 Module |
| Zusatzmodul | - Stromquelle fuer seriellen Bus<br>- Anschluss fuer Parametrier- und Strukturiereinheit |

Das an der TH Ilmenau entwickelte System S 4000 wird im KEAW durch den VEB Wetron Weida zur Produktion ueberfuehrt.

Literatur

/1/ Schlott, St.: Softwarearchitektur eines modularen Mehrrechnersystems der dezentralen Prozessdatenverarbeitung. Dissertation A, TH Ilmenau, 1986

/2/ Fengler, W. u. a.: Modulares Mikrorechnersystem MMS 881. Forschungsbericht, TH Ilmenau, Sektion TBK, 1986

/3/ Faerber, G. u. a.: Bussysteme. Regelungstechnische Praxis 26 (1984), H 5, 6

/4/ PROMAR 5000 - Sprachbeschreibung. EAW-elektronik, Berlin, 1985

/5/ TGL 44590/01: Werkzeugmaschinen mit speicherprogrammierbaren Steuerungen. Karl-Marx-Stadt, 1985

/6/ Brodie, L.: Starting FORTH, Prentice-Hall, 1982

/7/ Kowalski, O. u. a.: Monitore fuer Systeme der Prozessdatenverarbeitung. Technischer Bericht, TH Ilmenau, 1987

/8/ Fengler, W.; Roth, M.: Einchip-Rechner-Schaltkreise. Mikroprozessortechnik 1 (1987), H. 2, S. 37-41

/9/ Pflichtenheft zur Erzeugnisentwicklung, Stufe K1 VEB Wetron Weida, F. u. E., 26. 4. 88

# ПРОЕКТИРОВАНИЕ МИКРОПРОЦЕССОРНЫХ МОДУЛЕЙ РАСПРЕДЕЛЕННЫХ СИСТЕМ УПРАВЛЕНИЯ ТЕХНОЛОГИЧЕСКИМ ОБОРУДОВАНИЕМ

Жданов, В.С., д.т.н., профессор
Гапонов, А.А., аспирант
Никитин, Н.М., старший научный сотрудник
Окунев, С.Л., к.т.н., ассистент
Московский институт электронного машиностроения, Москва, СССР

Основой создания распределенных систем управления (РСУ) являются управляющие микропроцессорные модули (УМ), построенные на основе недорогих, надежных малогабаритных средств вычислительной техники - микропроцессоров и микро-ЭВМ. Управляющие микропроцессорные модули нижнего уровня выполняют первичную обработку информации и локальное управление отдельными единицами оборудования, УМ верхних уровней осуществляют оптимизацию и управление процессом в целом (на основе координации работы УМ локального управления), обеспечивают связь с оператором системы /I/. В зависимости от сложности решаемой задачи и требований к времени реакции определяются такие параметры УМ как: состав и структура УМ, быстродействие, разрядность, объем памяти, мощность интерфейса ввода-вывода и т.д. Для успешного функционирования в составе РСУ каждый УМ должен иметь средства сопряжения, обеспечивающие взаимодействие УМ путем обмена данными. Средства сопряжения УМ должны обеспечить возможность объединения в единую систему разнородных по интерфейсам, быстродействию и программному обеспечению УМ с учетом заданных интенсивностей обмена и объемов передаваемой информации.

Анализ принципов построения РСУ, существующих систем данного класса и тенденции их развития /I,2,3/ показывает, что одним из ведущих направлений является создание РСУ в виде моноканальной локальной управляющей сети. Моноканал обеспечивает полносвязность системы, но исключает возможность одновременного взаимодействия более чем одной пары устройств, подключенных к нему. Использование моноканала, который может иметь различную топологию, физическую среду передачи данных и скорость передачи, обеспечивает гибкость структуры РСУ, возможность модификации, развития системы, но вместе с тем требует решения задач, связанных с множественным доступом к моноканалу. Средства сопряжения должны

обеспечить требуемые функциональные характеристики РСУ, важнейшей из которых является время ответа, то есть время от момента поступления сообщения для передачи до момента окончания передачи сообщения. С точки зрения разработчика системы важным является эффективность использования ресурсов УМ и средств сопряжения, определяемая их загрузкой. Значительное число параметров средств сопряжения и стохастический характер взаимодействия УМ не позволяют дать строгую математическую постановку и решение задачи проектирования в общем виде, поэтому задача проектирования решается путем сведения её к задаче анализа различных вариантов, то есть "синтезу через анализ". В этих условиях наиболее эффективным решением данной задачи является создание семейства математических моделей отдельных компонент УМ и системы в целом на принципах функциональной и иерархической декомпозиции моделей. В этом случае при разработке УМ и его средств сопряжения из большого числа возможных вариантов выбирается ряд наиболее приемлемых решений, которые затем анализируются с учетом большего числа факторов на основе более детальных и сложных моделей. Последовательное проведение этой процедуры позволяет выбрать техническое решение, удовлетворяющее требованиям к системе, при этом модели большей сложности используются для анализа сравнительно небольшого числа вариантов построения средств сопряжения, что повышает эффективность моделирования.

Контроллеры сопряжения, входящие в состав УМ РСУ, обеспечивают взаимодействие УМ через моноканал и реализуют функции двух нижних уровней эталонной модели взаимодействия открытых систем /4/. По функциональному признаку в контроллере сопряжения можно выделить два основных блока: блок сопряжения с моноканалом и блок сопряжения с внутренней шиной УМ. Блок сопряжения с моноканалом обеспечивает формирование кадра заданного формата, контроль правильности принятой из моноканала информации, преобразование информации из параллельного кода в последовательный и обратно, передачу/прием информационных кадров и т.п. Важнейшей функцией данного блока является реализация метода множественного доступа моноканала. Блок сопряжения с внутренней шиной УМ обеспечивает взаимодействие с центральным процессором УМ, буферизацию принимаемой и передаваемой информации. При заранее определенном интерфейсе между блоком сопряжения с моноканалом и блоком сопряжения с внутренней шиной УМ конкретный тип и параметры каждого из

блоков могут определяться в значительной мере независимо /5/.
В качестве модели взаимодействия УМ используются модель в виде замкнутой многоканальной трехфазной системы массового обслуживания (СМО). Обслуживание на первой фазе СМО соответствует генерации сообщений управляющими модулями, обслуживания на второй фазе СМО соответствует реализации метода множественного доступа блоком сопряжения с моноканалом. Обслуживание на третьей фазе СМО соответствует передаче сообщений по моноканалу.
Блоки сопряжения с внутренней шиной УМ вносят задержку при приеме/передаче сообщений, связанную с буферизацией сообщений, реализацией доступа к внутренней шине микропроцессорного модуля. Недостаточное число буферных узлов и неоптимальный способ управления ими могут приводить к потерям сообщений. Анализ различных вариантов построения блоков сопряжения с внутренней шиной УМ позволил выделить две основные структуры таких блоков: с использованием буфера типа "первый вошел - первый вышел" и несколькими каналами прямого доступа к памяти; с использованием нескольких буферных узлов и нескольких каналов прямого доступа к памяти. Для оценки функциональных характеристик блоков указанного типа используются аналитические модели в виде многоканальных СМО с ограниченным временем ожидания и ограниченным числом мест в очереди. Модели указанных типов позволяют оценить задержки в обработке и вероятность потери сообщений при заданных интенсивностях и объемах поступающих сообщений.
В качестве модели функционирования УМ используется СМО с переключаемыми очередями. При этом дисциплина обслуживания заявок и параметры обслуживающего прибора определяются структурой УМ. Использование аналитических моделей при разработке УМ и средств сопряжения имеет ряд ограничений, связанных с предположением о простейшем входном потоке, сложностью получения функциональных характеристик РСУ при неоднородных потоках сообщений. Поэтому были разработаны имитационные модели отдельных компонент УМ и средств сопряжения. Возможность совместного использования указанных моделей при анализе всей системы определяется тремя основными условиями: согласованием моделей по входу-выходу, единой формой представления обслуживаемых заявок, синхронизацией работы моделей единым модельным временем.
Указанные аналитические и имитационные модели реализованы программно на языке ФОРТРАН и функционируют под управлением ОС ДОС

КП на мини-ЭВМ СМ-4. Разработанные программные средства были использованы при создании ряда РСУ.

ЛИТЕРАТУРА:


1. Антропов М.Б., Шенброт И.М. Комплексы средств для построения АСУ технологическими процессами. - Автоматика и телемеханика, 1985, № 9, с. 5-21.
2. Горностаев Ю.Н., Дрожжинов В.И. Сетевая интеграция автоматизированного машиностроительного производства на базе протоколов MAP/TOP.-М.:МЦНТИ, 1987. 87с.
3. Жданов В.С., Саксонов Е.А. Микропроцессорные устройства локальной автоматики в распределенных системах управления технологическим оборудованием. - В кн.: Применение ЭВМ в технике и управлении производством.: Материалы УШ Междунар. конф. COMPCONTROL 87, 4.П/2.-М., 1987, с.162-164.
4. Взаимосвязь открытых систем /ВОС/: новые международные стандарты по архитектуре и протоколам систем распределенной обработки информации.- Тематический выпуск. ТИИЭР, 1983, том 71, № 12.
5. Жданов В.С., Окунев С.Л., Саксонов Е.А. Применение микропроцессорных управляющих устройств в системах управления ГАП. - В кн.: Проблемы создания гибких производственных систем и роль ГАП при внедрении "безлюдной" технологии в промышленности: Тезисы докл. Всесоюзн.н.-т. конференции.-Гродно.-М.: Радио и связь, 1986, с.74.

# РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ЛОКАЛЬНЫХ МИКРОПРОЦЕССОРНЫХ ДИНАМИЧЕСКИХ РЕГУЛЯТОРОВ

Жданов В.С., проф., д.т.н., МИЭМ, Москва, СССР,
Булгаков М.В., доц., к.т.н., МИЭМ, Москва, СССР;
Ижванов Ю.Л., доц., к.т.н., МИЭМ, Москва, СССР

Успехи, достигнутые в последние годы в области микропроцессорной техники, привели к бурному развитию распределенных локальных систем автоматического управления и цифровых систем первичной обработки информации. Удобство работы с цифровыми сигналами, гибкость цифровых систем управления, широкие возможности перестройки структуры, высокая точность и надежность работы - вот лишь немногие из достоинств таких систем. Однако, главное преимущество распределенных микропроцессорных локальных систем управления в том, что они позволяют реализовать гораздо более широкий спектр алгоритмов регулирования и обработки информации, чем аналоговые. Это обстоятельство приводит к тому, что центр тяжести при разработке локальных микропроцессорных систем управления перемещается на разработку программного обеспечения таких систем, как наиболее трудоемкий элемент проектирования. Естественным шагом в направлении решения этой проблемы является создание САПР локальных цифровых САУ. Существующие в настоящее время САПР САУ позволяют решать широкий круг задач, связанных с анализом и синтезом автоматических регуляторов ( в том числе и цифровых ), однако ни в одной из известных САПР такого типа решение задачи не доводится до уровня готовых программ, ориентированных на использование конкретных микропроцессорных средств заданной конфигурации, взаимодействующих между собой с использованием определенной локальной вычислительной сети. Это резко снижает эффективность использования САПР САУ, поскольку специалист в области автоматического регулирования на определенном этапе разработки фактически исключается из процесса проектирования.

В докладе обсуждаются проблемы, возникающие при разработке САПР программного обеспечения локальных цифровых систем управления технологическим оборудованием и систем оперативной первичной обработки информации.

Исходными данными, используемыми для решения задачи, являются:
- математическая модель объекта управления, заданная в виде дифференциальных, конечно-разностных и логико-динамических уравне-

ний в терминах вход-выход или в терминах пространства состояний; экспериментально снятые характеристики поведения объекта в режиме нормальной эксплуатации или под воздействием тестовых сигналов; характеристики действующих на объект управления возмущений и шумов измерения; структурная схема объекта управления.

- цель управления, формулируемая в виде задач минимизации квадратичного функционала, оптимального быстродействия, заданного расположения корней характеристического уравнения, задач терминального управления.
- ограничения на значения управляющих воздействий и фазовых координат системы.
- конфигурация и используемых технических средств и их характеристики (адреса, допустимые для использования в ОЗУ и ПЗУ, адреса регистров внешних устройств и т.п.).
- периодичность и номенклатура передаваемой на верхний уровень управления информации (текущих значений технологических параметров, их статистических характеристик, сигналов о выходе значений параметров за допустимые пределы и др.).

В состав АРМ проектировщика программного обеспечения локальных цифровых систем управления входит мини ЭВМ "Электроника 100/25", микро ЭВМ ДВК-3, блок управления "Орион-3", аналоговые стенды или АВМ. Программы комплекса функционируют под управлением операционной системы RT-11 и реализуются на ЭВМ ДВК-3. Мини ЭВМ, функционирующая под управлением операционной системы TSX, служит для хранения файлов базы данных и решения задач, требующих значительных вычислительных ресурсов. Если пользователя удовлетворяют возможности микро ЭВМ ДВК-3, в частности, если файлы базы данных допустимо хранить на гибких дисках, мини ЭВМ может быть исключена из состава комплекса. Количество однотипных технических средств на каждом уровне комплекса может варьироваться в широких пределах.

Программное обеспечение комплекса состоит из 6 подсистем.

I. Подсистема работы с базой данных (специализированная СУБД). Эта подсистема обеспечивает ввод, вывод, хранение и корректировку информации о структуре объекта управления и регулятора, их передаточных функциях и/или уравнениях в пространстве состояний, матриц коэффициентов квадратичного критерия качества и т.п.

2. Подсистема моделирования. Эта подсистема обеспечивает цифровое моделирование основных управляющих и возмущающих воздействий, а также моделирование поведения объекта и замкнутой системы управления под действием этих сигналов Эта же подсистема обеспечивает линеаризацию и дискретизацию уравнений объекта управления.
3. Подсистема расчета характеристик. Эта подсистема обеспечивает вычисление времени переходного процесса, перерегулирования, значений квадратичного критерия качества. Эта же подсистема обеспечивает построение частотных и амплитудно-фазовых характеристик системы управления, вычисление статистических характеристик сигналов.
4. Подсистема расчета параметров регулятора. Эта подсистема обеспечивает расчет параметров цифровых регуляторов оптимальных по квадратичному критерию качества, регуляторов оптимальных по быстродействию, модальных регуляторов.
5. Подсистема отладки программ регулирования. В функции подсистемы отладки входит автоматическая загрузка программы цифрового регулирования по линии связи ИРПС в блок управления "Орион", обмен сообщениями в соответствии с стандартным протоколом между блоком управления и микро ЭВМ ДВК-3, занесение результатов полунатурного моделирования в базу данных.
6. Подсистема визуализации результатов моделирования.

Результатом решаемой в диалоговом режиме задачи проектирования является программное обеспечение системы локального регулирования заданным технологическим объектом, получаемое в виде программного модуля в формате абсолютной загрузки, и предназначенного для прошивки в ПЗУ или автоматической загрузки в ОЗУ блоков управления, оснащенных процессорами с системой команд LSI-11. Инвариантная компонента управляющей программы включает монитор, осуществляющий временную диспетчеризацию, и подпрограммы (драйверы) внешних устройств для ввода и вывода аналоговых и цифровых сигналов. Прикладными компонентами управляющей программы являются подпрограммы выработки уставки, подпрограммы динамического регулирования, программы логического управления по дискретным входам и выходам, а также программы обмена информацией с ЭВМ вышестоящего уровня.

# ENTWURFSMETHODIK UND SPRACHUNTERSTÜTZUNG FÜR VERTEILTE STEUERUNGEN IN DER NACHRICHTENTECHNIK

Hübner,U., Prof.Dr.sc.techn., KDT, Karl-Marx-Stadt, DDR

## 1. Einführung

Mittlere und große Vermittlungssysteme für alle Arten digitaler Informationen werden heute als verteilte Mehrrechnersteuerung realisiert. Die Software wird damit zum funktionsbestimmenden Teil des Systems.
Komplexität und Gesamtaufwand solcher Softwareentwicklungen sind hoch, man rechnet mit hunderten oder tausenden Mannjahren. Aus diesem Grund sowie wegen einiger spezieller Randbedingungen ist es gerechtfertigt, für derartige Softwareentwicklungen Methoden, Sprachen und Hilfsmittel bereitzustellen, die optimal auf die zu lösende Aufgabe abgestimmt sind.

Bei "eingebetteten Echzeitsteuerungen" kann das Steuerungssystem nicht direkt zur Entwicklung der Software verwendet werden.
Im Hinblick auf eine lange Nutzungsdauer und Hardwareunabhängigkeit wurden die Hilfsmittel fuer die Betriebssysteme WEGA/MUTOS /1/ entwickelt. Neben dem Vorteil der Verfügbarkeit weitgehend kompatibler Betriebssysteme auf praktisch allen Rechnerklassen und der (voraussichtlich) längsten Lebenserwartung unter den heute bekannten Betriebssystemfamilien steht damit auch eine optimal auf den Einsatzfall "Softwareentwicklung" abgestimmte Basis zur Verfügung.
Alle Werkzeuge sind selbst in C (bzw. darauf aufbauenden Grammatik-Definitionssprachen) implementiert.

In der Nachrichtentechnik müssen (strenger als auf anderen Gebieten) internationale Standardisierungen beachtet werden, da die Fernmeldenetze weltweit zusammenarbeiten müssen. Wegen der notwendigen Koexistenz von Einrichtungen aus vielen Technikgenerationen ist die Orientierung auf Standards aber auch national wichtig. Für die Belange der Software sind folgende CCITT-Empfehlungen von Interesse /2/:

Z.100-Z.104: SDL - Specification and Description Language
(Sprache für Spezifizierung, Entwurf und Beschreibung von Systemen)

Z.200: CHILL - CCITT High Level Language
(höhere Echtzeit-Programmiersprache)

Z.300-Z.3xx: MML - Mensch-Maschine-Sprache (Vorschriften zur Dialoggestaltung für Bedienterminals...)

Diese Spachen sind außerhalb der Nachrichtentechnik noch wenig verbreitet; einige der verwendeten Mittel und Methoden sind aber auch bei anderen Projekten verteilter Echtzeitsteuerungen nutzbar.

Der Entwicklungsablauf (Bild 1) und die dazu benutzte Umgebung kann ah verschiedenen Aspekten orientiert werden /3/:

- Methoden
- formale Sprachen
- "Werkzeugkasten" ...

Diese Aspekte durchdringen sich gegenseitig, d.h. die Sprache bestimmt wesentlich die Art und Weise des "Denkens"; Methoden sind ohne Werkzeuge kaum durchsetzbar ...

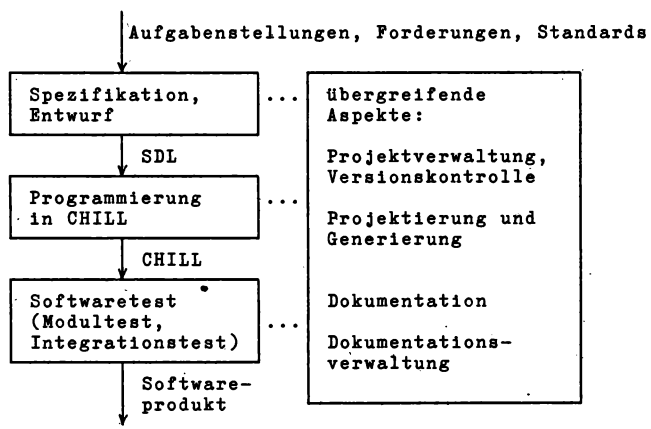



Bild 1: Entwicklungsablauf (vereinfacht)

## 2. Spezifikations- und Beschreibungssprache SDL

Die Spache SDL erlaubt die grafische Darstellung der Aufteilung eines großen Systems in Subsysteme und weiter verfeinert schließlich in Prozesse; dabei können die Informationsbeziehungen zwischen den Systemkomponenten ausgedrückt werden (Bild 2).

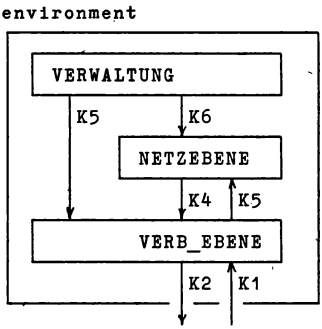



Bild 2: SDL/GR-Beispiel

Auf der Ebene der Prozesse können Abläufe auf der Basis eines (erweiterten) Zustandsmodells dargestellt werden. SDL eignet sich einmal zur Darstellung des Systemverhaltens gegenüber seiner Umgebung (Soll- und Istverhalten), aber auch zur Beschreibung interner Abläufe auf verschiedenenen Detaillierungsstufen.

## 3. Programmiersprache CHILL

Diese Programmiersprache wurde Ende der siebziger Jahre im Rahmen der CCITT-Standardisierung entwickelt. Es handelt sich um eine relativ moderne höhere Programmiersprache mit Prozeßkonzept. Aus heutiger Sicht gibt es natürlich noch weitere Sprachen, die im gleichen Zeitraum entstanden sind und die die zu stellenden Anforderungen aus technischer Sicht erfüllen könnten (z.B. Modula-2, ADA, PLZ/RTC). Da aber nicht nur technische Gründe hier eine Rolle spielen, basiert die Softwaretechnologie der Nachrichtentechnik auf CHILL.

Die Sprachdefinition ist sehr umfangreich (was wie bei ADA

schon einige Kritik hervorgerufen hat /4/), der Standard läßt aber die Bildung von Untermengen und implementierungsspezifische Ergänzungen in gewissem Rahmen zu. Davon wurde Gebrauch gemacht, um die Handhabbarkeit der Sprache zu verbessern und um Lücken in der Sprache (insbesondere bei der Anwendung in verteilten Systemen) zu schließen /5/.

CHILL-Prozesse können dynamisch erzeugt und beendet werden. Zu einem Prozeßkörper können mehrere gleichartige ablaufende Vorgänge (in verschiedenen Bearbeitungszuständen) existieren. Um eine Mehrdeutigkeit des Begriffs "Prozeß" zu vermeiden, bezeichnet man den ablaufenden Vorgang als "Prozeßinkarnation". Die Prozesse kommunizieren untereinander über ein asynchrones Botschaftenkonzept (Bild 3).

```
/* Deklaration des Kommunikationsobjekts: */
signal BOTSCHAFT = (int,char) to EMPFAENGER;

SENDER: process;
        send BOTSCHAFT (7,'X');
end SENDER;

EMPFAENGER: process;
  dcl X1 int; X2 char;
        receive case
        (BOTSCAFT in X1, X2): /* Aktionen ... */
        (TIMEOUT):            /* Aktionen ... */
        esac;
end EMPFAENGER;
```

Bild 3: Beispiel zur CHILL-Prozeßkommunikation

Literatur:


/1/ Claßen,L.;U.Oefler: UNIX und C
VEB Verl. Technik, Berlin, 1987

/2/ CCITT: Red book, Vol. VI; Genf, 1985

/3/ Dart,S.A.;R.J.Ellison;P.H.Feiler;A.H.Habermann:
Software Development Environments
Computer 20(1987)11, S.18-28

/4/ Young,S.J.: Real Time Languages: Design and Development
John Wiley & Sons, Chicester, 1982

/5/ Hübner,U.: CHILL-Programmiersystem für die Nachrichtentechnik. Nachrichtentechnik Elektronik 38(1988)5, S.188-190

# U840PC - SPEZIALPROZESSOR FÜR BINÄRE SPEICHERPROGRAMMIERBARE STEUERUNGEN

Fleischmann, G., Dipl.-Ing., VEB MME /FEE-2
Krumbein, F., Dipl.-Ing., VEB MME /FEE-2

## 1. Kurzcharakteristik

Der unipolare Schaltkreis U840PC ist ein Spezialprozessor für binäre speicherprogrammierbare Steuerungen. Seine Architektur ist auf die Unterstützung steuerungsspezifischer Besonderheiten sowie auf die problemorientierte Programmierung anhand der in der Steuerungstechnik üblichen bitorientierten Beschreibungsmittel ( Programmablaufgraph, Automatengraph, BOOLEsche Gleichungen, Logikplan usw.) zugeschnitten. Zur Kommunikation mit Standardprozessorsystemen bietet der U840PC Busanforderungs- und Bestätigungssignale, U855- kompatible Handshake-Steuersignale und Befehle für blockweisen Datentransfer von und zu externen Speichern. Damit ergeben sich zwei Einsatzmöglichkeiten:

- als selbständiger Spezialprozessor zur logischen Verknüpfung von Einzelbitsignalen und Verwaltung von Software-Zeitgliedern,
- als Coprozessor für Einzelbitverarbeitung und Datenaufbereitung in peripheren Ein- Ausgabe- Einheiten von Standardprozessor-Systemen.

Gehäuse: Plast-Chip-Carrier PLCC64, 1.27 mm Pinraster
Technologie: komplementäre Silicon-Gate-Technologie (CSGT)
Programmspeicher: extern, Standard-ROM (EPROM / RAM)
ext. Adressbereich: 4 kByte, auf 64 kByte erweiterbar
Takt: Einphasentakt (extern = intern), fmax= 2.5 MHz (4.0 MHz)
Betriebsspannung: UDD = 5 V +/- 0.25 V
Betriebsstrom: IDD < 15 mA / f = 2.5 MHz
Eingangsspannung: HIGH: UIH $\geq$ 2.4 V
LOW : UIL $\leq$ 0.8 V
Ausgangsspannung: HIGH: UOH $\geq$ UDD - 0.8 V / -IOH = 1.6 mA
LOW : UOL $\leq$ 0.4 V / IOL = 1.6 mA
pull-down-Eingangsstrom: 5 uA < IPD < 25 uA
Interne Baugruppen:

- 64 Register (8 Bit)
- vier Ein-Ausgabe-Ports (je 8 Bit, interne pull-down-Widerst.)

- Zähler/Timer (8 Bit) mit Vorteiler (8 Bit)
- BOOLEsche Verarbeitungseinheit (1 Bit: NEGATION, AND, OR, ANTIVALENZ, ÄQUIVALENZ, acht Klammerebenen, Syntaxtest)
- Byte-Verarbeitungseinheit (8 Bit: Inc., Dec., Laden und Vergleich mit Konstante, Registertransfer/-vergleich )
- serielle Ein-Ausgabe-Einheit (2 x 8 Bit, kundenspezifisch)
- Datenbuseinheit (8 Bit)
- Programmzähler (16 Bit)
- Adreßverarbeitungseinheit (16 Bit)
- Mikroprogrammsteuerwerk

## 2. Arbeitsweise

Zur Instruktionsverarbeitung wird beim U840PC das Pipeline-Prinzip verwendet. Das grundsätzliche Aufnahmeprinzip externer Ereignisse an den E/A- Ports ist die zyklische Abfrage (Polling), die durch die Abarbeitung des "IO"- Befehls vollzogen wird. Die Reaktionszeit des U840PC auf äußere Ereignisse an den E/A-Ports ist softwareabhängig und gleich der Zykluszeit, die sich aus dem zeitlichen Abstand der Ausführung der "IO"- Befehle ergibt. Entsprechend der Programmierung arbeitet der U840PC in der prozessorientierten oder zyklischen Betriebsart.
Die Datenmanipulation vollzieht sich im internen RAM des U840PC (Register R0...R63). Die E/A- Ports liegen im Adreßraum des internen RAM (R0...R3) und besitzen Master-Slave-Verhalten. Während der Berechnung des aktuellen Zyklus erfolgt keine Änderung der Ausgangsbelegung des Prozessors und keine Übernahme einer neuen Eingangsbelegung. Dies geschieht erst mit der Abarbeitung eines "IO"- Befehls. Der "IO"- Befehl ist ein Ein- Byte- Befehl ohne Adreßinformation, d.h. die Kommunikation an den E/A- Ports wird ohne Beanspruchung des Datenbusses an allen E/A- Pins gleichzeitig durchgeführt (Datenausgabe vor Datenübernahme).
Für Havarie- und Not- Aus-Situationen besitzt der U840PC einen maskierbaren Interrupteingang. Der Interrupt (Aufruf eines Unterprogramms auf einer Festadresse) stellt beim U840PC ein zusätzliches Aufnahmeprinzip externer Ereignisse dar, besitzt eine höhere Priorität als das Polling (Busanforderung: höchste Priorität) und wird unabhängig vom Polling verarbeitet.

## 3. Erweiterungsprinzip

Die Anzahl der verfügbaren Ein- und Ausgänge des U840PC ist um jeweils 32 E/A- Pins modular erweiterbar, indem unter Ausnutzung des Prinzips der dezentralen Befehlsdekodierung und -verwaltung weitere U840PC als Erweiterungsprozessoren EPU (engl. Extended Prozessing Unit) an den Datenbus und den Steuerbus angeschlossen werden. Damit ergeben sich folgende Vorteile:

1. Die Hardware ist einfach realisierbar. Es existiert keine übergeordnete zentrale Steuereinheit, es sind in Abhängigkeit der Erweiterung nur Treiber notwendig, und das EPU- System arbeitet wegen der dezentralen Befehlsdekodierung und E/A- Verwaltung bei gleicher Frequenz schneller als ein zentral gesteuertes System mit erweiterter Anzahl von E/A- Bauelementen.
2. Die Software ist unkompliziert. Sie enthält keine Algorithmen zur Verwaltung und Synchronisation des EPU- Systems. Bei der modularen E/A- Erweiterung wird der Software-Ingenieur nur mit einem erweiterten prozessorinternen Datenadreßraum konfrontiert, der im Befehlssatz des U840PC berücksichtigt ist. Im EPU- System entfallen Maßnahmen und Befehle für den Datentransfer zwischen E/A- Bauelementen und einer zentralen Verarbeitungseinheit.

Im Vergleich zu einem Mehrprozessorsystem ist der Nachteil zu verzeichnen, daß ein EPU- System trotz mehrerer Prozessor-Bauelemente nur einen Programmspeicher enthält, d.h. ein (erweitertes) Einprozessorsystem darstellt. Damit sind nicht verschiedene Programme parallel abarbeitbar.

## 4. Befehlssatz

Der Befehlssatz des U840PC enthält 64 Basisbefehle, 5 Adressierungsarten und besitzt eine Register-zu-Register-Struktur.
Die Befehle unterteilen sich in:

- 20 Befehle für die Bit- Verarbeitung, (s. Tafel 1.)
- 10 Befehle für die Byte- Verarbeitung (s. Abschnitt 1.)
- 15 Befehle für die Pointer- Verwaltung,
- 8 Befehle für Programmverzweigungen und Unterprogrammtechnik,
- 5 Befehle für Ein-Ausgabe und blockweisen Datentransfer,
- 4 Steuer- und sonstige Befehle.

Tafel 1. Mnemonik der bitverarbeitenden Befehle des U840PC

| NR | Befehl | Byte | Takte | Zeichen | Bemerkungen |
|---|---|---|---|---|---|
| 1 | JV0 | 2 | 5/8 | B -0- | Bit-Test und bedingter Sprung |
| | | 3 | 7/8 | | bei Bit = "0" |
| 1 | JV1 | 3 | 7/10 | B -1- | Bit-Test und bedingter Sprung |
| | | 4 | 9/10 | | bei Bit = "1" |
| 3 | RES | 1/2 | 2/4 | B:== 0 | RESET Bit |
| 4 | SET | 1/2 | 2/4 | B:== 1 | SET Bit |
| 5 | AV | 1/2 | 2/4 | V | AND Variable |
| 6 | AVN | 1/2 | 2/4 | ./V | AND, NEGATION Variable |
| 7 | OV | 1/2 | 2/4 | + V | OR Variable |
| 8 | OVN | 1/2 | 2/4 | +/V | OR, NEGATION Variable |
| 9 | EQV | 1/2 | 2/4 | * V | ÄQUIVALENZ Variable |
| 10 | EQVN | 1/2 | 2/4 | */V | ÄQUIVALENZ, NEGATION Variable |
| 11 | ATV | 1/2 | 2/4 | # V | ANTIVALENZ Variable |
| 12 | ATVN | 1/2 | 2/4 | #/V | ANTIVALENZ, NEGATION Variable |
| 13 | GV | 1/2 | 2/4 | = V | GLEICH (Zuweisung) Variable |
| 14 | GVN | 1/2 | 2/4 | =/V | GLEICH, NEGATION Variable |
| 15 | AKA | 1 | 2 | ( | AND, Klammer auf |
| 16 | OKA | 1 | 2 | + ( | OR, Klammer auf |
| 17 | EQKA | 1 | 2 | * ( | ÄQUIVALENZ, Klammer auf |
| 18 | ATKA | 1 | 2 | # ( | ANTIVALENZ, Klammer auf |
| 19 | KZ | 1 | 2 | ) | Klammer zu |
| 20 | KZN | 1 | 2 | )/ | Klammer zu, NEGATION |

(Der Begriff "Variable" symbolisiert auch den nachfolgenden Term entsprechend den Regeln der BOOLEschen Algebra. Die Doppelangaben bei TAKT und BYTE beziehen sich auf unterschiedliche Adressierungsarten bzw. bedingte Befehlsausführungen.)

Literatur: "Handbuch U840PC" (Diskette)
Applikation: VEB MME, Abt. FEA3, Tel. Erfurt 51076 App. 40

ASSEMBLER/DEBUGGER FÜR U840PC AUF PERSONALCOMPUTER

Grimm, W. ,Dipl. Ing., VEB MME /FEE 2

## 1. Einleitung

Zur Unterstützung der Programmierung des Spezialprozessors U840PC wird vom Hersteller Entwicklungssoftware bereitgestellt Diese Software ist auf allen Computern lauffähig auf denen TURBO - PASCAL (Version 3.0) implementiert ist.

## 2. Hauptprogramm

Das Hauptprogramm dient zur Steuerung der einzelnen Programm - aufrufe Hier erfolgt die Eingabe des aktuellen Workfiles. Folgende Programmodule können aufgerufen werden.

Editor
Assembler
Debugger
Plot
Bit

## 3. Editor

Zur effektiven Arbeit bei der Programmerstellung und Programm - testung wurde in das Programmsystem ein Texteditor implementiert Die Befehle wurden an den Texteditor WORD STAR angelehnt Dabei sind nur solche Befehle verwendet worden , die unbedingt für die Texteingabe und Korrektur notwendig sind Im folgenden sind die einzelnen Editorbefehle aufgeführt:

Cursor rechts,links,auf,ab
Seite vorwärts,rückwärts
Finde String,next String
lösche Zeichen auf und vor dem Cursor
lösche Zeile

```
Einfügen von Zeilen,Zeichen
Exit
Quit
```

Auf Grund der verwendeten Datenstruktur kann der Editor Text - dateien von beliebiger Länge verarbeiten.

## 4. Assembler

Der Assembler hat die Aufgabe, den Quelltext auf Syntaxfehler zu überprüfen und einen Zwischenkode für den Debugger zu erzeugen . Oie Syntax der einzelnen Befehle entspricht der im Handbuch U840 veröffentlichten Mnemonik . Zur effektiven Programmarbeit wurde neben speziellen Assemblerbefehlen noch die Möglichkeit der Arbeit mit Variablen und symbolischen Adressen geschaffen Der Assembler unterstüzt die Möglichkeiten der direkten und indirekten Register-adressierung des U840PC . Bei der Programmübersetzung ist es möglich, die Befehlsoptimierung zu aktivieren . Die Wirkung der Befehlsoptimierung wird im nachfolgenden Beispiel erläutert .

; Mit Semikolon wird eine Kommentarzeile gekennzeichnet.

```
        xl      s1,r3,b4    ; ein Minuszeichen in einer Zeile gibt an
                            ; daß eine  Variablendeklaration erfolgt
ml:     set     xl          ; ml: ist eine Sprungmarke
        jp      ml              ; unbedingter Sprung  zur  Marke ml
        ld      x2, low  (ml)   ; lade  Low-Teil  der  Adresse von ml
        ld      x3, high (ml)   ; lade  High-Teil der  Adresse von ml
        byte    %h27            ; Speicherplatz wird mit hex 27 belegt
        byte    12              ; Speicherplatz wird mit dec 12 belegt
        ram     64              ; Reservierung von 64 Byte RAM-Bereich
                                ; dies ist fuer den IN  und OUT Befehl
                                ; von Bedeutung
        loc %h64                ; Setzen  des Adresszeigers auf hex 64
```

Programmbeispiel vor der Befehlsoptimierung:

```
        av      sl,r34,b4       ;Zwei-Byte-Befehl
        ov      sl,r34,b5       ;Zwei-Byte-Befehl
m1:     gv      sl,r34,b2       ;Zwei-Byte-Befehl
        avn     sl,r34,b5       ;Zwei-Byte-Befehl
        ovn     sl,r35,b3       ;Zwei-Byte-Befehl
        eqvn    sl,r35,b7       ;Zwei-Byte-Befehl
        jp0     m2
```

Nach der Befehlsoptimierung:

```
        av      sl,r34,b4       ;Zwei-Byte-Befehl
        ov      b5              ;Ein -Byte-Befehl
m1:     gv      sl,r34,b2       ;Zwei-Byte-Befehl
        avn     b5              ;Ein -Byte-Befehl
        ovn     sl,r35,b3       ;Zwei-Byte-Befehl
        eqvn    b7              ;Ein -Byte-Befehl
        jp0     m2
```

Im obigen Programmbeispiel bewirkt die Befehlsoptimierung eine Verkürzung des Programms um 3 Byte

5. Debugger

Der Debugger stellt das Kernstück des Programmpakets dar Damit besteht die Möglichkeit einer gezielten Testung des Programms Dazu werden in der oberen Bildschirmhälfte der Quelltext und in der unteren Bildschirmhälfte alle Register des Schaltkreises mit ihrem aktuellen Wert angezeigt Dem Nutzer stehen eine Reihe von Befehlen zur Verfügung , mit denen er in der Lage ist sehr einfach und schnell Fehler im Programm zu lokalisieren

Befehlsliste:

| | |
|---|---|
| step | schrittweise Programmabarbeitung; |
| run | kontinuierliche Programmabarbeitung; |

| | |
|---|---|
| break | Setzen eines Breakpoints auf eine Programmzeile oder Marke, (max 8 Breakpoints); |
| watch | Beobachtung von Bits oder Bytes, das Programm wird angehalten, wenn sich der Wert ändert (max 8 Watchpoints ); |
| pattern | Aktivierung eines Patternfiles zur Dateneingabe über die Ports; |
| go | Start des Programms ab Adresse; |
| set | Änderung der Darstellungsart des Debuggers; |
| deposit | Änderung der Registerinhalte; |
| examine | Darstellung der Registerinhalte; |
| help | Beschreibung der einzelnen Befehle; |
| list | Listen von Programmbereichen; |
| status | Darstellung des Debuggerstatus; |
| cancel | löschen von Break und Watchpoint; |
| ctrl Y | Programmabbruch, falls Anwenderprogramm in endloser Schleife, Rückehr in den Debugger; |
| ctrl I | Auslösung eines U840PC - Interrupts; |

Neben der Programmabarbeitung erfolgt noch eine umfangreiche Fehlerbehandlung Sie garantiert, daß unzulässige Betriebs - zustände des Schaltkreises erkannt werden können.

## 6. PLot

Das Plotterprogramm wird benötigt , wenn nach der Abarbeitung von Patternfiles die Ergebnisse ausgewertet werden sollen. Mit ihm können ausgewählte Ein - Ausgänge in übersichtlicher Form auf dem Bildschirm oder dem Drucker ausgegeben werden.

## 7. Ausgabe der ROM - Belegung

Mit diesem Programmteil ist es möglich, das übersezte Programm für die Programmierung im EPROM aufzubereiten Hierbei sind ver - schiedene Ausgabeformen entsprechende der Anwenderforderung möglich.